Towards High Resolution Numerical Algorithms for Wave Dominated Physical Phenomena
AFOSR FA9550-05-1-0473

Final Report

T. Warburton

January 30, 2009

Contents

20120918091

1 Introduction

This is the final report for AFOSR FA9550-05-1-0473. As the project developed a clear theme emerged from the work we undertook: increasing the efficiency of discontinuous Galerkin time-domain(DGTD) solvers for wave propagation, in particular for solving time-dependent electromagnetic field propagation. We developed four main strategies:

- **Multi-Rate Time Stepping DGTD**: Decreasing the number of time steps needed for a simulation by allowing the solution in different parts of the problem domain to advance at different rates dictated by local resolution density.
- **CFL Enhanced High-Order DGTD**: Developing a new filter that increases the maximum allowable time steps for high-order simulations.
- **Complete Radiation Boundary Conditions**: reating a new family of radiation boundary conditions that gracefully truncate the solution domain close to scattering objects with efficient reduction of spurious reflection even in the presence of evanescent fields.
- **GPU Hardware Accelerated DGTD**: Accelerated computation by using the latest generation of general purposes graphics processing units (GPGPU) as computational platforms. This is a natural prototype to adapt DGTD to the trend of computing towards many-core devices.

The project also took an interesting detour: designing algorithms for creating massive tilings of polyomino pieces as suitable templates for passive array antennae production.

2 The Discontinuous Galerkin Time Domain Method

To simplify presentation of the advances achieved during this project we discuss them in the context of homogeneous material, non-dimensionalized Maxwell's equations being solved in a domain Ω with a cover $\Omega = \bigcup_{k=1}^{k=K} D^k$ of K elements that are typically but not limited to simplices, tensor-product elements or even general polygons and polyhedra. The assumption of homogeneous material is only necessary for optimal performance of high-order filtering described later in this report.

In terms of the electric field $\mathbf{E}$, the magnetic field $\mathbf{H}$, the charge density ρ, the current density j, the permittivity ϵ, and the permeability μ, Maxwell's equations are:

$$\frac{\partial \mathbf{D}}{\partial t} - \nabla \times \mathbf{H} = -j, \qquad \frac{\partial \mathbf{B}}{\partial t} + \nabla \times \mathbf{E} = 0, \tag{1}$$

$$\nabla \cdot \mathbf{D} = \rho, \qquad \nabla \cdot \mathbf{B} = 0. \tag{2}$$

The semi-discrete DG variational equations for Maxwell's equations demand that we find $\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{B} \\ \boldsymbol{D} \end{pmatrix} \in X^h \times Y^h$ such that in the k'th element:

$$\begin{aligned} \left(\phi, \frac{\partial \boldsymbol{B}}{\partial t} + \nabla \times \boldsymbol{E}\right)_{D^k} &= \left(\phi, -\boldsymbol{n} \times (\boldsymbol{E}^* - \boldsymbol{E}^-)\right)_{\partial D^k}, \\ \left(\psi, \frac{\partial \boldsymbol{D}}{\partial t} - \nabla \times \boldsymbol{H}\right)_{D^k} &= \left(\psi, +\boldsymbol{n} \times (\boldsymbol{H}^* - \boldsymbol{H}^-)\right)_{\partial D^k}. \end{aligned}$$

for all $\begin{pmatrix} \phi \\ \psi \end{pmatrix} \in X^h \times Y^h$. We have allowed the variational spaces for the solution fields to be quite general, but in practice we use product spaces of broken polynomial spaces on each element:

$$X^h = Y^h = \bigoplus_{k=1}^{k=K} \left(P^N\left(D^k\right)\right)^3.$$

Furthermore, we did not specify the extension states $\boldsymbol{H}^*, \boldsymbol{E}^*$ for the distributional derivatives. These are typically set to be an average or upwind combination of the '-' boundary trace of the solution from within the k'th element and the '+' boundary trace of the solution from the elements sharing parts of the boundary of the k'th element, ∂D^k. This formulation is detailed in the first book on discontinuous Galerkin methods [34].

2.1 GPGPU Accelerated Implementations of the DGTD

Discontinuous Galerkin methods [34,38,49] are, at first glance, a rather curious combination of ideas from Finite-Volume and Spectral Element Methods. Up close, they are very much high-order methods by design. But instead of perpetuating the order increase like conventional global methods, at a certain level of detail, they switch over to a decomposition into computational elements and couple these elements using Finite-Volume-like surface Riemann solvers. This hybrid, dual-layer design allows DGTD to combine advantages from both of its ancestors. But it adds a third advantage: By adding a movable boundary between its two halves, it gives implementers an added degree of freedom when bringing it onto computing hardware.

At the same time, computing is undergoing a transformative and disruptive change at the moment: Previously, the execution time of a given code could be determined simply by counting how many floating point operations it executes. More recently, memory bottlenecks, in the form of bandwidth limitation and fetch latency, have taken over as the dominant factor, and CPU manufacturers use large amounts of silicon to mitigate this effect. It is quite instructive and somewhat depressing to compare the chip area used for caches, prediction, and speculation in recent CPUs to the area taken up by the actual functional units. The picture is changing, however, and graphics processors, having recently turned into general-purpose programmable units, were the first to do away with expensive caches and combat latency by massive parallelism instead. In this research, we explore how and with what benefit DGTD can be brought onto these graphics processing units (GPUs).

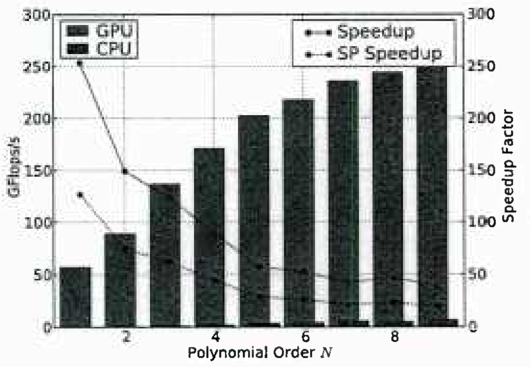
We addressed two challenges as part of this research: First, how should the computational work be partitioned? In a distributed-memory setting, the answer is quite naturally domain decomposition. For GPUs, it turns out, there are several possibilities, and there is often no single answer that works well in all settings. Second, DGTD implementations on serial processors often rely heavily on the availability of off-the-shelf, pre-tuned linear algebra and communication primitives. These aids are either unavailable or unsuitable on a GPU platform, and in stark contrast to the relatively straightforward implementation of DGTD on serial machines, optimal use of graphics hardware for DGTD presents the implementor with a staggering number of choices. We will describe these choices as well as a generative approach that exploits them to adapt the method to both the problem and the hardware at run time.

Using graphics processors for computational tasks is by no means a new idea. In fact, even in the days of marginally programmable fixed-function hardware, some (especially particle-based) methods obtained large speedups from running on early GPUs. (e.g. [46]) In the domain of solvers for partial differential equations, Finite-Difference Time-Domain (FDTD) methods are a natural fit to graphics processors and obtained speedups of about an order of magnitude with relative ease. (e.g., [45]) Finite Element solvers were also brought onto GPUs relatively early on (e.g., [40]), but often failed to reach the same impressive speed gains observed for the simpler FDTD methods. While high-level abstractions such as BrookGPU [36] have made relatively complex computations on GPUs possible, only the recent release of compute abstractions that are less encumbered by their graphics heritage [47,48] has interested many more researchers in bringing increasingly challenging computations onto the GPU (e.g. [41]). Taking advantage of these recent advances, this paper presents, to the best of our knowledge, one of the first general finite-element based solvers that achieves more than an order of magnitude of speedup on a single graphics processor when compared to a CPU implementation of the same method.

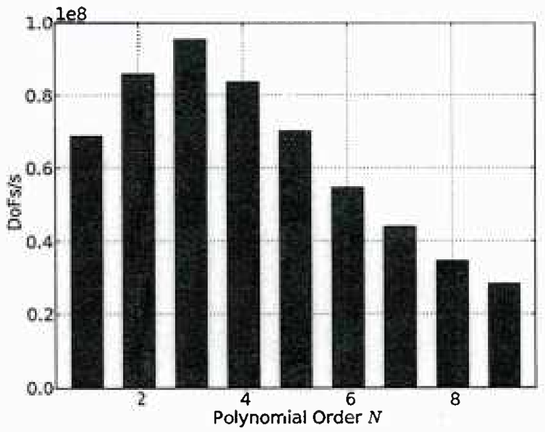
A sizable part of this speedup is owed to our use of high-order approximations. High-order methods require more work per degree of freedom than low-order methods. This increased arithmetic intensity shifts the method from being limited by memory bandwidth further being limited by compute bandwidth. The relative abundance of cheap compute bandwidth on a GPU makes high-order methods especially beneficial there. Figure 1(a) portrays the speed of our solver in comparison with a double-precision CPU implementation running on a single core of a 3 GHz Intel Core2 Duo E8400 CPU using ATLAS 3.8.2 [54] for its element-local operations. The results are scaled as floating point operations per second, obtained by counting the number of floating point additions and multiplications in the algorithm and dividing by the time in seconds. GPU times were measured using the `cuEventElapsedTime()` call. Factoring in a potential CPU disadvantage of a factor of two for the use of double precision, the GPU outperforms the CPU by factors ranging from 19.4 at order nine to 127 at order one. At the practically relevant orders of three and four, the adjusted speedup factors are 62 and 44, respectively.

Orders three and four are particularly favorable not only for their appreciable speedups and their moderate time step requirements [53]. They also achieve the peak nodal value throughputs on the GPU as shown in Figure 1(b). Naturally, high-order approximations of solutions to partial differential equations contain much more information per DOF than do solutions obtained via low order methods. This is most apparent in the number of DOFs required to accurately represent one wavelength [42]. Interestingly, we observe that DGTD methods of orders one and two do not appear to conform as well to the hardware's granularities and therefore achieve lower overall throughput than the next higher ones. This crossover between granularity effects and the increase in floating point work with growing N makes DGTD methods of orders three and four the fastest DGTD methods on a GPU even on a per-DOF basis.

It is interesting to correlate the achieved floating point bandwidth of each computational component from Figure 2(a) with the bandwidth reached for transfers between the processing core and global memory, shown in Figure 2(b). We obtained these numbers by counting the number of bytes fetched from global memory

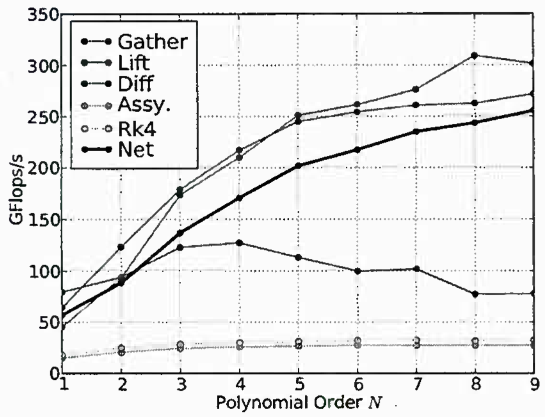


(a) Discontinuous Galerkin performance in GFlops/s on a GPU (using single precision) and a CPU (using double precision). Speedup factors are given both for direct comparison and under the assumption that the CPU doubles its speed when computing in single precision.

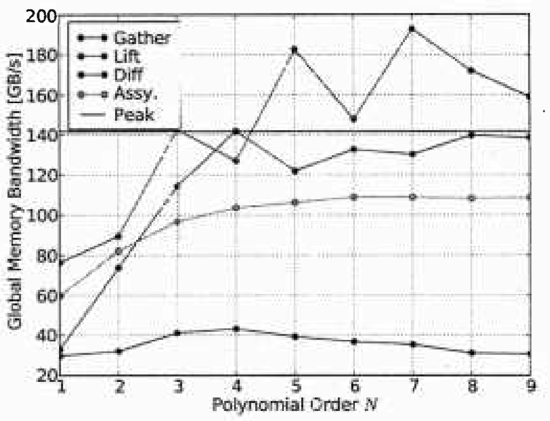


(b) Degrees of freedom processed per second on the GPU.

Figure 1: Performance characteristics of DGTD on Nvidia graphics hardware.



(a) Compute bandwidth in GFlops/s achieved by each part of the DGTD operator, at various polynomial orders. The published theoretical peak floating point performance for the hardware on which these tests were run is 933 GFlops/s [55].



(b) Memory bandwidths in GB/s achieved by each part of the DGTD operator. The peak memory bandwidth published by the manufacturer is 141.7 GB/s. Values exceeding peak bandwidth are believed to be due to the presence of a texture cache.

Figure 2: Performance characteristics of DGTD on Nvidia graphics hardware, continued.

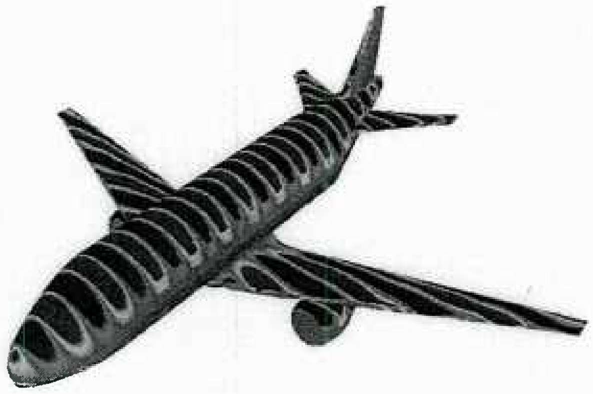


Figure 3: A sample high-order DGTD scattering simulation computed using a GPU accelerated workstation in minutes.

either directly or through a texture unit. The theoretical peak memory bandwidth published by Nvidia is 141.7 GB/s, shown as a black horizontal line. Perhaps the most striking feature at first is the fact that the memory bandwidth measured for flux lifting transcends this theoretical peak at orders five and above.

2.1.1 Summary: Accelerated GPGPU Computing with DGTD

We have adapted a number of pre-existing DGTD codes for the GPU, enabling a thorough comparison of strategies for mapping the method onto the hardware. A final code was written that combines the insights gained from its precursors.

We found that, using our strategies, high-order DGTD methods can reach double-digit percentages of published theoretical peak performance values. **For a suitable CPU-limited workload, this means that a single workstation equipped with multiple GPUs can do work that previously required a roomful of computing hardware**. Alternatively, a cluster of machines equipped with these cards can run simulations that were previously outside the reach of all but the largest supercomputers. This lets the size and complexity of simulations that researchers can afford on a given hardware budget jump by more than an order of magnitude. To allow potential users of our work to take advantage of this jump, and to let them easily reproduce and improve on our results, we have made our code available under the terms of the GNU General Public License. It may be downloaded from `http://www.dam.brown.edu/people/kloeckner/hedge`. A simple version with less flexibility can be downloaded from `http://www.caam.rice.edu/~timwar/RMMC`.

In future work will extend the approach to make use of double precision floating point support that has become available on recent Nvidia hardware. In addition, we would like to broaden the applicability of our methods by exploring their use for nonlinear conservation laws as well as elliptic problems.

Many-core computing presents a rare opportunity, and we feel that discontinuous Galerkin methods have a number of unique characteristics that make them unusually suitable for many-core platforms. In the past, users have chosen low-order methods because of the perceived expense involved in running simulations at a high order of accuracy. While this perception was questionable even in the past, we feel that many-core architectures disproportionately *favor* high order and significantly drive down its relative cost. Moreover, unlike most other numerical schemes for solving partial differential equations, DGTD methods make the order of accuracy a tunable parameter. These factors combine to give the user a maximum of control over both performance and accuracy. Furthermore, with the advent of quad-gpu workstations it is now possible for engineers to perform relatively large, high-order, 3D DGTD electromagnetic simulations in a routine manner in situ.

3 Local Time Stepping for Discontinuous Galerkin Time Domain

There is a growing literature of local time stepping methods for discontinuous Galerkin (DG) methods that allow for variable rates of time stepping throughout the computational domain (see for instance [27–32]). We consider here an alternative and simpler class of multi-rate Adams-Bashforth time stepping methods originally proposed for stiff/non-stiff systems of ODEs by Gear and Wells [33]. Combining the multi-rate Adams-Bashforth time stepper with the DG space discretizations allows for flexible domain discretization in space with full non-conformimg h and p adaptivity in space and more importantly local time steps can be chosen uniquely in each element. One particular goal is to avoid the problem of having to use artificially small global time steps to handle elements produced in mesh generation that are degenerately small.

After choosing a basis for the test and trial space we obtain semidiscrete equations for the unknown solution $\boldsymbol{Q}_N = \begin{pmatrix} \boldsymbol{H} \\ \boldsymbol{E} \end{pmatrix}$, given by $\frac{d\boldsymbol{Q}}{dt} = \mathfrak{L}\boldsymbol{Q}$, where $\mathfrak{L}$ corresponds to a matrix representing the spatial derivatives discretized with the discontinuous Galerkin distributional derivatives.

If all elements were to advance with the same dt we might choose an Adam-Bashforth (AB) time integrator then the fully discrete equations are

$$\boldsymbol{Q}^{n+1} = \boldsymbol{Q}^n + dt\left(a_0\mathfrak{L}\boldsymbol{Q}^n + a_1\mathfrak{L}\boldsymbol{Q}^{n-1} + a_2\mathfrak{L}\boldsymbol{Q}^{n-2}\right),$$

with coefficients for the first to third order AB schemes given in Table 1.

Table 1: Coefficients for Adams-Bashforth time integrator: $\boldsymbol{a}$ and for the first half and second half time step Adams-Bashforth time integrators: $\boldsymbol{b}$ and $\boldsymbol{c}$.

Order	a_0	a_1	a_2	b_0	b_1	b_2	c_0	c_1	c_2
1	1	0	0	1	0	0	1	0	0
2	$\frac{3}{2}$	$-\frac{1}{2}$	0	$\frac{5}{4}$	$\frac{-1}{4}$	0	$\frac{7}{4}$	$\frac{-3}{4}$	0
3	$\frac{23}{12}$	$-\frac{16}{12}$	$\frac{5}{12}$	$\frac{17}{12}$	$\frac{-7}{12}$	$\frac{2}{12}$	$\frac{29}{12}$	$\frac{-25}{12}$	$\frac{8}{12}$

Invoking basic properties of the polynomial bases we can estimate that there is an upper limit for the time step given by $dt < C\frac{h}{N^2}$ where the constant depends on the order of the AB scheme. Here h is the minimum element size of *all* the elements in the mesh covering the domain. Unfortunately this is a global restriction and dt may be made small by a tiny minority of elements with high aspect ratio or small size driven by small scale domain geometry features.

To remedy the global time step restriction we consider using a multirate extension to the basic AB schemes. We consider a partition of the domain into "coarse" and "fine" parts denoted by Ω_C and Ω_F respectively. Denoting the solution restricted to the fine and coarse domains as $\boldsymbol{Q}_F$ and $\boldsymbol{Q}_C$ respectively and reordering the degrees of freedom we write

$$\frac{d}{dt}\begin{pmatrix} \boldsymbol{Q}_F \\ \boldsymbol{Q}_C \end{pmatrix} = \begin{pmatrix} \mathfrak{L}_{FF} & \mathfrak{L}_{FC} \\ \mathfrak{L}_{CF} & \mathfrak{L}_{CC} \end{pmatrix}\begin{pmatrix} \boldsymbol{Q}_F \\ \boldsymbol{Q}_C \end{pmatrix}$$

For simplicity we assume here that the largest stable time step for the coarse mesh, dt, is twice as large as the largest stable time step for the fine mesh, dt_F. We now describe the multirate AB scheme in stages. We first time march the fine mesh by $dt/2$ with

$$\boldsymbol{Q}_F^{n+1/2} = \boldsymbol{Q}_F^n + \frac{dt}{2}\begin{pmatrix} \mathfrak{L}_{FF}\left(a_0\boldsymbol{Q}_F^n + a_1\boldsymbol{Q}_F^{n-1/2} + a_2\boldsymbol{Q}_F^{n-1}\right) \\ +\mathfrak{L}_{FC}\left(b_0\boldsymbol{Q}_C^n + b_1\boldsymbol{Q}_C^{n-1} + b_2\boldsymbol{Q}_C^{n-2}\right) \end{pmatrix}. \quad (3)$$

where the b coefficients are appropriate for integrating the quadratic interpolant of regularly spaced data over t^n to $t^{n+1/2}$. Next we integrate the fine mesh for the second $dt/2$

$$Q_F^{n+1} = Q_F^{n+1/2} + \frac{dt}{2}\left(\begin{array}{c} \mathcal{L}_{FF}\left(a_0 Q_F^{n+1/2} + a_1 Q_F^n + a_2 Q_F^{n-1/2}\right) \\ +\mathcal{L}_{FC}\left(c_0 Q_C^n + c_1 Q_C^{n-1} + c_2 Q_C^{n-2}\right) \end{array} \right), \tag{4}$$

where the c coefficients are appropriate for integrating the quadratic interpolant of regularly spaced data over $t^{n+1/2}$ to t^{n+1}. Finally, we update the coarse mesh

$$Q_C^{n+1} = Q_C^n + dt \left(\begin{array}{c} \mathcal{L}_{CC}\left(a_0 Q_C^n + a_1 Q_C^{n-1} + a_2 Q_C^{n-2}\right) \\ +\frac{1}{2}\mathcal{L}_{CF}\left(a_0 Q_F^n + a_1 Q_F^{n-1/2} + a_2 Q_F^{n-1}\right) \\ +\frac{1}{2}\mathcal{L}_{CF}\left(a_0 Q_F^{n+1/2} + a_1 Q_F^n + a_2 Q_F^{n-1/2}\right) \end{array} \right). \tag{5}$$

Combining these equations we obtain

$$Q^{n+1/2} = Q^n + \frac{dt}{2}\mathcal{L}\left(\begin{array}{c} a_0 Q_F^n + a_1 Q_F^{n-1/2} + a_2 Q_F^{n-1} \\ b_0 Q_C^n + b_1 Q_C^{n-1} + b_2 Q_C^{n-2} \end{array} \right), \tag{6}$$

and for the second substep

$$Q^{n+1} = Q^{n+1/2} + \frac{dt}{2}\mathcal{L}\left(\begin{array}{c} a_0 Q_F^{n+1/2} + a_1 Q_F^n + a_2 Q_F^{n-1/2} \\ c_0 Q_C^n + c_1 Q_C^{n-1} + c_2 Q_C^{n-2} \end{array} \right). \tag{7}$$

We can also express the coarse-fine partition in terms of projections Π^C and Π^F

$$Q^{n+1/2} = Q^n + \frac{dt}{2}\mathcal{L}\left(\Pi^F\left(a_0 Q^n + a_1 Q^{n-1/2} + a_2 Q^{n-1}\right) + \Pi^C\left(b_0 Q^n + b_1 Q^{n-1} + b_2 Q^{n-2}\right)\right), \tag{8}$$

and for the second substep

$$Q^{n+1} = Q^{n+1/2} + \frac{dt}{2}\mathcal{L}\left(\Pi^F\left(a_0 Q^{n+1/2} + a_1 Q^n + a_2 Q^{n-1/2}\right) + \Pi^C\left(c_0 Q^n + c_1 Q^{n-1} + c_2 Q^{n-2}\right)\right), \tag{9}$$

We can further express each stage as a single step method:

$$\begin{pmatrix} Q^{n+1/2} \\ Q^n \\ Q^{n-1/2} \\ Q^{n-1} \\ Q^{n-2} \end{pmatrix} = A^1 \begin{pmatrix} Q^n \\ Q^{n-1/2} \\ Q^{n-1} \\ Q^{n-2} \end{pmatrix} \tag{10}$$

$$\begin{pmatrix} Q^{n+1} \\ Q^{n+1/2} \\ Q^n \\ Q^{n-1} \end{pmatrix} = A^2 \begin{pmatrix} Q^{n+1/2} \\ Q^n \\ Q^{n-1/2} \\ Q^{n-1} \\ Q^{n-2} \end{pmatrix} \tag{11}$$

where

$$A^1 = \left(\begin{array}{c|c|c|c} I + \frac{a_0 dt}{2}\mathcal{L}\Pi^F + \frac{b_0 dt}{2}\mathcal{L}\Pi^C & \frac{a_1 dt}{2}\mathcal{L}\Pi^F & \frac{a_2 dt}{2}\mathcal{L}\Pi^F + \frac{b_1 dt}{2}\mathcal{L}\Pi^C & \frac{b_2 dt}{2}\mathcal{L}\Pi^C \\ \hline \boldsymbol{I} & 0 & 0 & 0 \\ \hline 0 & \boldsymbol{I} & 0 & 0 \\ \hline 0 & 0 & \boldsymbol{I} & 0 \\ \hline 0 & 0 & 0 & \boldsymbol{I} \end{array}\right) \tag{12}$$

$$A^2 = \left(\begin{array}{c|c|c|c|c} I + \frac{a_0 dt}{2}\mathcal{L}\Pi^F & \frac{a_1 dt}{2}\mathcal{L}\Pi^F + \frac{c_0 dt}{2}\mathcal{L}\Pi^C & \frac{a_2 dt}{2}\mathcal{L}\Pi^F & \frac{c_1 dt}{2}\mathcal{L}\Pi^C & \frac{c_2 dt}{2}\mathcal{L}\Pi^C \\ \hline \boldsymbol{I} & 0 & 0 & 0 & 0 \\ \hline 0 & \boldsymbol{I} & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & \boldsymbol{I} & 0 \end{array}\right) \tag{13}$$

A necessary condition for stability is to find dt such that $\boldsymbol{A}^2\boldsymbol{A}^1$ has spectral radius less than one as we can combine these two steps to yield

$$\begin{pmatrix} Q^{n+1} \\ Q^{n+1/2} \\ Q^{n} \\ Q^{n-1} \end{pmatrix} = \boldsymbol{A}^2\boldsymbol{A}^1 \begin{pmatrix} Q^{n} \\ Q^{n-1/2} \\ Q^{n-1} \\ Q^{n-2} \end{pmatrix}. \tag{14}$$

Furthermore, for eigenvalues with unit value we must determine that the corresponding eigenvector spaces are non-degenerate. We are in the process of analyzing these conditions. In the meantime we have conducted numerical experiments. In Figure 4 we verify for a specific instance that the approach does in fact allow for doubling of time-steps using the multirate time-stepping method outlined above.

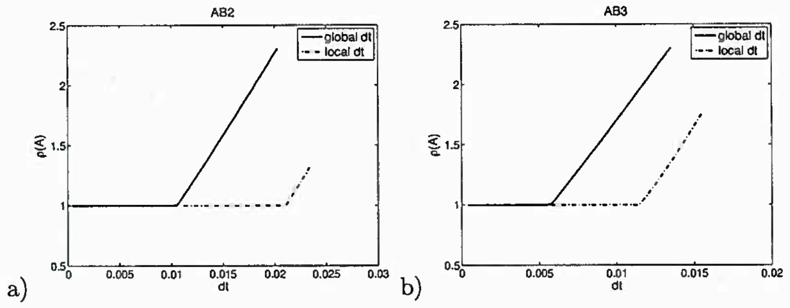


Figure 4: Maximum eigenvalue of the one-step system matrix for a) AB2, b) AB3 using global and two-level local time stepping. Notice that the spectral radius exceeds 1 at approximately twice the dt for the local time stepping method than for the global time stepping method.

Transition I: We communicated a variant of this time-stepping formulation to Adour Kabakian, of Hypercomp. He implemented a version and presented results at the EMCC Annual Meeting, May 2008. We have included some snapshots of results he presented at this meeting. In particular results for local time stepping for the slit plate test case where speed ups of 2.5 are achieved, see Figure 5.

3.1 Modeling Point Sources

As we began to instrument the DGTD simulation code for localized forcing and monitoring of the solution we realized that there was an interesting issue in handling singular forcing while still maintaining high-order accuracy.

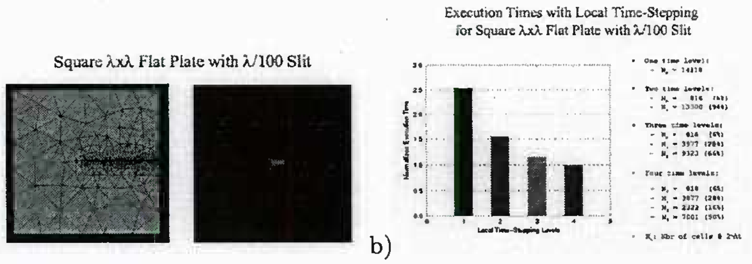


Figure 5: a) Highly refined mesh for card scatterer with slit. b) Table of timings depending on the number of levels used for time stepping. [Results courtesy of Adour Kabakian as presented at the EMCC Annual Meeting, May 2008.

Recalling the time dependent Maxwell's equations with an electric point source given in terms of a time modulated Dirac delta function centered at (x^s, y^s)

$$\begin{aligned} \frac{\partial \mathbf{D}}{\partial t} - \nabla \times \mathbf{H} &= -\mathbf{F}(t)\delta\left(x - x^s\right)\delta\left(y - y^s\right) \\ \frac{\partial \mathbf{B}}{\partial t} + \nabla \times \mathbf{E} &= 0, \end{aligned}$$

with corresponding DGTD weak form

$$\begin{aligned} \left(\phi, \frac{\partial \boldsymbol{B}}{\partial t} + \nabla \times \boldsymbol{E}\right)_{D^k} &= \left(\phi, -\boldsymbol{n} \times \left(\boldsymbol{E}^* - \boldsymbol{E}^-\right)\right)_{\partial D^k}, \\ \left(\psi, \frac{\partial \boldsymbol{D}}{\partial t} - \nabla \times \boldsymbol{H}\right)_{D^k} &= \left(\psi, +\boldsymbol{n} \times \left(\boldsymbol{H}^* - \boldsymbol{H}^-\right)\right)_{\partial D^k} + \left(\psi, \mathbf{F}(t)\delta\left(x - x^s\right)\delta\left(y - y^s\right)\right)_{D^k}. \end{aligned}$$

Our first attempt at handling the singular forcing involved using the selecting principle of the Dirac delta function to express the source term as a non-zero contribution in the element D^k that contains (x^s, y^s), given by

$$\left(\psi, \mathbf{F}(t)\delta\left(x - x^s\right)\delta\left(y - y^s\right)\right)_{D^k} = \mathbf{F}(t)\psi(x^s, y^s), \tag{15}$$

and zero in all other elements. In the context of the DGTD variational statement we are simply using an L^2 approximation of the Dirac delta. Unfortunately, this yields a slowly converging and highly oscillatory approximation to the singular function.

The second approach we used was to iteratively refine the element containing the singularity using the Bisection algorithm and then to use the Multi-Rate DGTD algorithm to locally time-step the small elements resulting from the refinement.

Our third approach to modeling the singular forcing is to use a narrow Gaussian as an approximation for the Dirac delta

$$\delta\left(x\right) \approx \frac{e^{-x^2/(\alpha dt)}}{\sqrt{\pi \alpha dt}}. \tag{16}$$

This can be resolved by the DGTD code when α is sufficiently large [resolution width comment].

The fourth approach we used to model the singular forcing relies on the linearity of Maxwell's equations. In this case we express the solution in the element containing the singularity as a linear combination of a regular part and the solution of the free space singularly forced problem i.e. $\mathbf{H} = \mathbf{H}^r + \mathbf{H}^s$ and $\mathbf{E} = \mathbf{E}^r + \mathbf{E}^s$. The solution to the free space problem can be calculated at relatively modest cost at the nodes on the boundary of the singularity containing element and its neighbors. In this way we do not have to explicitly use the singular forcing function within the variational forcing as a volume term, but rather as extra flux terms in the singularity containing element and its neighbors. For instance the variational equation in the singularity containing element will read

$$\begin{aligned}\left(\phi, \frac{\partial \boldsymbol{B}^r}{\partial t} + \nabla \times \boldsymbol{E}^r\right)_{D^k} &= \left(\phi, -\boldsymbol{n} \times \left(\boldsymbol{E}^* - \boldsymbol{E}^s - \boldsymbol{E}^{r,-}\right)\right)_{\partial D^k}, \\ \left(\psi, \frac{\partial \boldsymbol{D}^r}{\partial t} - \nabla \times \boldsymbol{H}^r\right)_{D^k} &= \left(\psi, +\boldsymbol{n} \times \left(\boldsymbol{H}^* - \boldsymbol{H}^s - \boldsymbol{H}^{r,-}\right)\right)_{\partial D^k}\end{aligned}$$

where the upwind fields have been computed with respect to the full fields. Notice how the source fields only appear in the surface fluxes. The key observation here is that we have entirely removed the singular components from the equations and this approach should guarantee optimal order convergence.

We are currently investigating the relative performance of each of these approaches.

4 Filtering High-Order Discontinuous Galerkin Time Domain for Efficiency

A significant numerical hurdle for wave propagation is the generic need for preservation of phase and amplitude information as rapidly varying wave profiles travel over long distances. Upwind DGTD has emerged as a competitive numerical modeling technique for achieving this goal. This method was originally proposed for neutron transport by Reed & Hill [23] and analysed by many including Lesaint & Raviart [21], Johnson & Pitkaranta [18], Richter [24], and Peterson [22]. DG was revisited for time-dependent conservation laws in a sequence of papers by Cockburn, Shu et al [6–8, 10]. The method is particularly attractive for wave propagation because of the ability to use high-order, piecewise polynomial approximations for the solution which provides an efficient mechanism to manage phase and dissipation errors (see for example Ainsworth [1]). The method has been shown to be effective on quadrilateral meshes by Kopriva et al [19, 20], on unstructured meshes of triangles with high-order approximation for Maxwell's equations by Hesthaven & Warburton [14, 15, 25], and on overlapping triangular meshes by Chung & Engquist [5]. Furthermore, this method automatically controls spurious solutions through selective dissipation of non-physical solutions (see for example Hesthaven & Warburton [16] and Warburton & Embree [26]). In addition, the structure of the numerical method results in computational algorithms which scale efficiently for large scale SIMD computations, Biswas et al [2].

The positive aspects of the upwind DGTD method are accompanied by a particular sour note. As the order of the polynomial approximation N is increased on a fixed mesh of elements, with mesh size h, the spatial differential operator discretized with DGTD typically has a spectral radius which grows as N^2/h. This is in contrast to typical finite difference discretized spatial differential operators which have spectral radii which grow as N/h. This translates into the reality that the DGTD method requires smaller time steps, dt, to scale its spatial derivative operator's spectrum into the stability region of a standard Runge-Kutta time-stepping method. Thus, the advantages of the DGTD method are tempered by the extra number of time steps compared with finite difference methods.

One goal of this project was to create a simple modification to the upwind DGTD method which reduces the spectral radius of a modified version of the discrete DGTD derivative operator to a more competitive N/h. By way of motivation we note the Hermite-Taylor method analyzed by Goodrich et al [12]. There it is shown that a combination of staggered Hermite interpolation and Taylor expansion based time stepping leads to a method with degree-independent time step stability constraints. Similarly, we find that for the case of an autonomous system of linear partial differential equations, combining the modified upwind DGTD spatial discretization with an $N+1$-stage, explicit, Runge-Kutta time-stepping method it is possible to choose stable time step size, dt, independent of the spatial polynomial order N. In the more general case of partial differential equations with nonlinear terms or time-dependent forcing, we can expect a factor proportional to N increase in the allowable time step size. In the conclusion we discuss the limitations of this approach.

4.1 Formulation

In this section we describe a prototypical upwind DGTD formulation for the one-dimensional, uni-directional, advection equation. We then describe how to introduce a covolume-based filter which allows for larger steps when time-stepping the discretized equations.

First introducing some notation: the spatial interval $\Omega = [a, b]$, is divided into a primal grid of K^1 elements, $T_h^1 = \bigcup_{k=1}^{k=K^1} I^{1,k}$, with $I^{1,k} = \left[x^{1,k}, x^{1,k+1}\right]$. We associate the following broken Sobolev spaces with the primal grid

$$H^m\left(\Omega, T_h^1\right) = \left\{q \in L^2\left(\Omega\right) : q|_{I^{1,k}} \in H^m\left(I^{1,k}\right) \text{ for } k = 1, ..., K^1\right\} , \tag{17}$$

where

$$H^m\left(I^{1,k}\right) = \left\{q \in L^2\left(I^{1,k}\right) : \sum_{\alpha=0}^{\alpha=m} \left\|\frac{\partial^\alpha q}{\partial x^\alpha}\right\|^2_{L^2(I^{1,k})} < \infty\right\} . \tag{18}$$

In each element the solution will be represented by polynomials of maximum degree N. The numerical solution on the primal grid is chosen from

$$V^{1,h} = \bigcup_{k=1}^{k=K^1} P^N\left(I^{1,k}\right) , \tag{19}$$

with no a priori assumption about the continuity of the the solution. We also require a covolume mesh whose elements are formed with end points in the interior of the primal grid. For definiteness we consider here the special case where these are chosen to be the midpoints of the subintervals of the primal grid, but as seen below such a precise choice is unnecessary. The covolume mesh then consists of K^2 elements, $\left\{I^{2,k}\right\}_{k=1}^{k=K^2}$, with elements $I^{2,k} = \left[\frac{x^{1,k-1}+x^{1,k}}{2}, \frac{x^{1,k}+x^{1,k+1}}{2}\right]$ for elements not on the boundary.

4.1.1 Upwind Discontinuous Galerkin Derivative Operator For Advection

We are interested in solving advection and wave type phenomena, and consequently consider the prototype one-way advection equation

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0 \text{ for } (x,t) \in [0,2) \times [0,1] , \tag{20}$$

with periodic boundary conditions $u(2,t) = u(0,t)$.

For a simplified presentation we consider the upwind discontinuous Galerkin discretization of this equation, because of its weighted residual formulation which has guaranteed stability, a block diagonal mass matrix and potentially high-order accuracy. The upwind DGTD variational statement involves finding $u \in V^{1,h}$ such that

$$\left(\phi, \frac{\partial u}{\partial t}\right)_{I^{1,k}} + \left(\phi, \frac{\partial u}{\partial x}\right)_{I^{1,k}} + \left(\phi, \frac{n_x - |n_x|}{2}[u]\right)_{\partial I^{1,k}} = 0 , \tag{21}$$

for all $\phi \in V^{1,h}$ and $k = 1, ... K^1$, where the notation $[u]$ denotes the trace of u from inside element k subtracted from the trace of u taken from within the element sharing a boundary vertex with element k. In familiar notation $[u] = u^+ - u^-$ where u^- is the internal trace of u in element k and u^+ is the external trace. Here n_x is the outwards facing normal at the element boundary and the boundary inner-product is defined by

$$(f,g)_{\partial I^{1,k}} = f\left(x^{1,k}\right) g\left(x^{1,k}\right) + f\left(x^{1,k-1}\right) g\left(x^{1,k-1}\right) \tag{22}$$

We introduce a slightly more general notation which allows us to choose between an upwind boundary treatment and a central treatment via the operator D^1_α, where $\alpha \in [0,1]$ is a parameter which interpolates between each type of treatment. Formally D^1_α is the operator which satisfies for any $u \in V^{1,h}$

$$(\phi, D^1_\alpha u)_{I^{1,k}} = \left(\phi, \frac{\partial u}{\partial x}\right)_{I^{1,k}} + \left(\phi, \frac{n_x - \alpha|n_x|}{2}[u]\right)_{\partial I^{1,k}}, \tag{23}$$

for $k = 1, ..., K^1$ and $\phi \in V^{1,h}$. Now we succinctly represent the upwind DGTD variational statement by

$$\left(\phi, \frac{\partial u}{\partial t}\right)_{I^{1,k}} + (\phi, D^1_\alpha u)_{I^{1,k}} = 0 \tag{24}$$

This notation highlights the significant fact that despite its apparent complexity, the DGTD operator is actually just the result of encoding a distributional derivative to account for boundary conditions introduced at the end points of each element.

After discretization of the $V^{1,h}$ variational space of functions defined with respect to the primal grid, the advection equation is reduced from a partial differential equation to an ordinary differential equation

$$\frac{d\mathbf{u}^1}{dt} = -\mathbf{D}^1_\alpha \mathbf{u}^1, \tag{25}$$

where $\mathbf{D}^1_\alpha$ is the upwind DGTD derivative matrix defined on the primal grid by

$$(\mathbf{D}^1_\alpha)_{ij} = \sum_k (\mathbf{M}^{1,k})^{-1}_{in}\left(\left(\phi_n, \frac{\partial \phi_j}{\partial x}\right)_{I^{1,k}} + \left(\phi_n, \frac{n_x - \alpha|n_x|}{2}[\phi_j]\right)_{\partial I^{1,k}}\right), \tag{26}$$

where $\mathbf{M}^{1,k}_{nm} = (\phi_n, \phi_m)_{I^{1,k}}$. It is useful to view the $\mathbf{D}^1_\alpha$ matrix as representing a discrete distributional derivative operator, and thus later on it will be transparent that modifying this operator by filtering matrices is not a complicated process. Once we establish the method, then we will appeal to the variational statement to establish stability but in terms of implementation it is most instructive to think in terms of he action of matrices on the solution.

We can immediately establish L^2 stability with the following result

$$\frac{1}{2}\frac{d}{dt}\sum_k (\mathbf{u}^1)^t \mathbf{M}^{1,k}\mathbf{u}^1 = -\frac{1}{2}\sum_k (\alpha|n_x|[u^1], [u^1])_{\partial I^{1,k}} \le 0, \tag{27}$$

which states succinctly that the choice of upwinding the solution at the end points of the elements before taking the DGTD distributional derivatives acts to reduce the L^2 energy of the numerical solution whenever there is a jump in the solution between elements.

We next introduce the discretized form of the transfer operators. These are the matrices $\mathbf{\Pi}^2$, $\mathbf{\Pi}^1$ defined by

$$\begin{aligned}(\phi^2_i, \phi^2_n)_{T^2_h}\mathbf{\Pi}^2_{nj} &= \sum_k \left((\phi^2_i, \phi^1_j)_{I^{1,k}\cap I^{2,k}} + (\phi^2_i, \phi^1_j)_{I^{1,k-1}\cap I^{2,k}}\right), & (28)\\ (\phi^1_i, \phi^1_n)_{T^1_h}\mathbf{\Pi}^1_{nj} &= \sum_k \left((\phi^1_i, \phi^2_j)_{I^{1,k}\cap I^{2,k}} + (\phi^1_i, \phi^2_j)_{I^{1,k+1}\cap I^{2,k}}\right), & (29)\end{aligned}$$

for all basis members $\{\phi^2_i\}_{i=1}^{i=N_p}$ on the K covolume mesh elements and all the basis members $\{\phi^1_i\}_{i=1}^{i=N_p}$ on the K primal grid elements. It is immediately observed that these transfer matrices are related by

$$\mathbf{M}^2\mathbf{\Pi}^2 = (\mathbf{M}^1\mathbf{\Pi}^1)^t. \tag{30}$$

4.1.2 Modified Upwind DGTD Method

We now have sufficient notation to propose a new type of DGTD method which we write in pseudo-code

```
1. Evaluate initial condition on primal grid.
2. Start time stepping, with an $s$-stage Runge-Kutta scheme.
3. For each RK stage:
    (a) $L^2$ project the solution from primal grid to covolume mesh.
    (b) Compute DGTD distributional derivative on covolume mesh, using upwind fluxes suitable for the
        advection equation.
    (c) $L^2$ project the DGTD distributional derivative to original grid.
    (d) Update solution for RK stage using $L^2$ projected DGTD distributional derivative.
4. Continue time stepping until final time reached.
```

We have added two projection stages. The first stage is designed to project the original piecewise polynomial solution on the primal grid onto the covolume mesh. The purpose of this projection is to diminish the maximum gradients achieved on the covolume mesh elements. We then evaluate the DGTD distributional gradient on the covolume mesh and project it back on to the primal grid, and again our aim is to reduce the potential for large near boundary gradients.

In matrix notation we are solving the following ordinary differential equation

$$\frac{d\mathbf{u}}{dt} = -\mathbf{\Pi}^1\mathbf{D}_\alpha^2\mathbf{\Pi}^2\mathbf{u}\,, \tag{31}$$

where the DGTD derivative matrix $\mathbf{D}_\alpha^2$ is defined on the covolume mesh. We can also relax the stabilization by averaging the stabilized DGTD derivative with the unstabilized DGTD derivative

$$\frac{d\mathbf{u}}{dt} = -\beta\mathbf{\Pi}^1\mathbf{D}_\alpha^2\mathbf{\Pi}^2\mathbf{u} - (1-\beta)\mathbf{D}_\alpha^1\mathbf{u} \tag{32}$$

where $\beta = 1 - O\left(\frac{1}{N}\right)$. It is straightforward to perform the L^2 stability analysis for this modified DGTD scheme, with the L^2 energy equation given by

$$\begin{aligned}\frac{d}{dt}\left\|u^1\right\|^2_{L^2(T_h^1)} &\leq -\frac{\beta}{2}\sum_k \left(\alpha|n_x|\left[\Pi^2 u^1\right],\left[\Pi^2 u^1\right]\right)_{\partial I^{2,k}} \\ &\quad -\frac{(1-\beta)}{2}\sum_k \alpha\left(|n_x|\left[u^1\right],\left[u^1\right]\right)_{\partial I^{1,k}}\,. \end{aligned} \tag{33}$$

Thus the numerical energy decreases if there are jumps in u^1 between elements on the primal grid and if there are jumps in $\Pi^2 u^1$ between elements on the covolume mesh. (We will subsequently observe that the filtering process creates an intermediate field with much reduced jumps.)

4.1.3 Motivation for Using a Covolume Mesh Based Filter

The traditional discontinuous Galerkin discretization of a first order distributional derivative has maximum eigenvalue which asymptotically grows as $O\left(\frac{N^2}{h}\right)$ when the order of polynomials used, N, is increased or h is decreased (e.g. [11, 13]). In the particular case of hp type methods, where it is considered desirable to use a high order polynomial approximation when possible, the derivative operators become artificially stiff.

This artificial stiffness is the result of the nature of the piecewise polynomial discretization spaces used. In this section we detail the causes behind the scaling and present a revealing observation that we use to build the modified DGTD methods.

We first recall the following polynomial inequalities (see for example Borwein and Erdelyi [3] for details and generalizations)

Theorem 4.1 (Bernstein's Inequality) *The inequality*

$$\left|\frac{dq}{dx}(x)\right| \le \frac{N}{\sqrt{1-x^2}} \|q\|_\infty \ , \quad -1 < x < 1$$

holds for every $q \in P^N([-1,1])$.

Theorem 4.2 (Markov's Inequality) *The inequality*

$$\left\|\frac{dq}{dx}\right\|_\infty \le N^2 \|q\|_\infty$$

holds for every $q \in P^N([-1,1])$.

We put these together and observe that

$$\left|\frac{dq}{dx}(x)\right| \le \min\left(N^2, \frac{N}{\sqrt{1-x^2}}\right) \|q\|_\infty \ , \quad -1 <= x <= 1 \ , \tag{34}$$

and since these estimates are sharp, the gradients of normalized n'th order polynomials may be up to $O\left(\frac{N^2}{h}\right)$ at the near boundary portion of the element and $O\left(\frac{N}{h}\right)$ near the center of the element. These inequalities explain the difference between the Courant-Friedrichs-Lewy (CFL) conditions of a typical finite difference method, where a moving interpolation stencil is used and slope information is generally extracted at element centers, and the DGTD method where slope information is used throughout.

As part of this effort we proved the following analogs that extend the above two inequalities to the case of the L^2 norm.

Lemma 4.3 (L^2 Version of Bernstein's Inequality) *The inequality*

$$\left\|\frac{dq}{dx}\right\|_{L^2([-a,a])} \le C_a N \|q\|_{L^2([-1,1])} \ , \tag{35}$$

where $C_a = \frac{\sqrt{3}+a}{1-a^2}$, *holds for every* $q \in P^N([-1,1])$.

Lemma 4.4 (L^2 Version of Markov's Theorm.) *Assume* $u \in P^N\left(I^{1,k}\right)$ *then*

$$\left\|\frac{dq}{dx}\right\|_{L^2([-1,1])} \le \sqrt{3} N^2 \|q\|_{L^2([-1,1])}$$

These results show that the pointwise non-uniformity of polynomial gradients also extends to L^2 and the underlying cause of the excess stiffness of DGTD is the piecewise polynomial approximation used. These results will also apply to FEM

We start with the combination of these two inequalities as our inspiration for using a co-volume grid based filter to smooth out the inhomogeneity of polynomial gradients. In Figure 6, we plot the upper bound (34) for three bi-unit one-dimensional elements which each overlap by half their lengths with their neighbors. This upper bound on the scaled gradient of polynomials grows dramatically near each of the

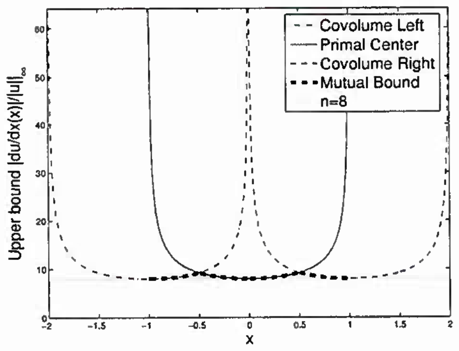


Figure 6: Combining upper bounds on the relative maximal gradient of polynomials in P^8 for primal and covolume mesh yields a more uniform bound.

element boundaries. However, we have also highlighted the minimum upper bound of all three in the central element and this curve is almost constant throughout this element. This observation yielded the possibility that we may be able to overcome the artificial stiffness induced in the DGTD scheme by the non-uniform gradients of polynomial approximations to fields. In the next sections we prove that covolume derivative operator, $\mathbf{\Pi}^1\mathbf{D}^2_\alpha\mathbf{\Pi}^2$, motivated by these inequalities, in fact satisfies the desired bound.

It is also necessary to study the properties of the projection-based filter that is implicit in our formulation. We note that these projection operators are both contractions, i.e. $\|\Pi^2u\| \leq \|u\|$ and $\|\Pi^1u\| \leq \|u\|$. Furthermore, we consider the action of the operator defined by $\Pi^2\Pi^1$. This operator is a filter and in subsequent sections we establish its properties by deriving bounds on $\|u - \Pi^2\Pi^1u\|$.

4.2 Bounds on the Covolume Derivative Operator

Here we present our main result on the N-dependence of the covolume derivative operator. We work in the framework described above periodically extending functions beyond $[0, 2)$.

Theorem 4.5 *Let q be a degree N piecewise polynomial function relative to the primal grid, $q \in V^{1,h}$ and set, for $\alpha \in [0, 1]$:*

$$\mathcal{D}_{S,\alpha}q = \Pi^1\mathbf{D}^2_\alpha\Pi^2q. \tag{36}$$

Then there exists a universal constant, C, such that:

$$\|\mathcal{D}_{S,\alpha}q\|_{L^2(\Omega)} \leq C\frac{N}{h}\|q\|_{L^2(\Omega)}. \tag{37}$$

As an immediate corollary we have:

$$\|\left(\beta\mathcal{D}_{S,\alpha} + (1-\beta)\mathbf{D}^1_\alpha\right)q\|_{L^2(\Omega)} \leq C\left(\beta\frac{N}{h} + (1-\beta)\frac{N^2}{h}\right)\|q\|_{L^2(\Omega)}, \tag{38}$$

so that the desired bound for the matrix appearing in (32) is obtained if we take $\beta = 1 - O\left(\frac{1}{N}\right)$.

Having established the improved stability property, we also established a suboptimal error bound for the proposed covolume method. We note that, in comparison with error estimates for the standard approach, our estimates are $O(h^{1/2})$ worse. We suspect that, in fact, the proposed method is as accurate as the standard one. We show below that this is true in the constant coefficient case on uniform grids; then we can use Fourier methods as in [17] to show that the error is $O(h^{N+1})$. Our numerical experiments indicate that the modified method is also as accurate as the standard method for nonuniform grids. However, direct application of the method of proof from, e.g., [9] fails. We plan to consider alternative analyses in the future.

To simplify the presentation we consider the $\beta = 1$ fully filtered case, as the extension to the partially filtered case is straightforward. We also only consider the semidiscrete scheme.

Theorem 4.6 *Suppose u satisfies (20) with $u(x,0) = u_0(x) \in H^{r+1}$. Then for all $0 \le s \le \min(n,r)$ there exist constants $C_{n,s}$ such that the solution, u_h, of the semidiscrete problem satisfies the error estimate:*

$$\begin{aligned} \|u-u_h\|_{L^2(\Omega)} \le\ & C_{n,s} \sum_{k=1}^{k=K^1} \left(h^{1,k}\right)^{s+1} \|u\|_{H^{s+1}(I^{1,k})} \\ & + C_{n,s} \sum_{k=1}^{k=K^1} \left(h^{1,k}\right)^{s} T \max_{t\in[0,T]} \left(\|u\|_{H^{s+1}(I^{1,k})} \right) \\ & + C_{n,s} \sum_{k=1}^{k=K^2} \left(h^{2,k}\right)^{s+1} \|u\|_{H^{s+1}(I^{2,k})} \\ & + C_{n,s} \sum_{k=1}^{k=K^2} \left(h^{2,k}\right)^{s} T \max_{t\in[0,T]} \left(\|u\|_{H^{s+1}(I^{2,k})} \right), \end{aligned}$$

These estimates can be improved in the specific case when the domain is periodic and the mesh consists of equal sized elements.

4.3 Numerical Results

In this section we investigate the sharpness of the analytical prediction of order of accuracy, and in fact go further and experimentally test the spectral correctness of the discrete modified upwind DGTD operators. This latter test is an important indicator of potential solution quality and robustness in practical situations. Admittedly, the method as presented is rather limited in geometric flexibility and in representing solutions of low regularity but it is important to establish the potential for the method before we should even consider further development.

4.3.1 Test 1: Advection on Regular Grid

In our first test we chose to discretize the periodic interval $[0,2]$ with 20 elements using 7'th order polynomials in each element. In Figure 7 we show the full spectra of the discretized upwind DGTD differentiation operator for $\beta = 0, 0.2, 0.4, 0.6, 0.8, 1.0$. It is immediately apparent that the modified operator has a much reduced spectral radius as β increases.

However, it is also apparent that the filtering process is wrapping the spectrum tightly near the imaginary axis so we must be concerned about the possibility of spurious eigenvalues for the discrete operator. Also in Figure 7 we show the same spectral plots but each zoomed to the limits of the $\beta = 1$ case.

We see that with $\beta = 1$ the normally highly dissipated DGTD eigenmodes have reduced dissipation and are now in close proximity to the imaginary axis, in particular near the origin, and may present in time-domain simulations as parasitic solutions. This effect can be mitigated in this specific case by choosing $\beta = 0.8$ say. Then there is a clear separation between the spurious and non-spurious eigenvalues. There is clearly a need for a subtle balance between decreasing the spectral radius of the original DGTD operator and confusing the spectrum near the imaginary axis. The well represented Fourier modes evident on the imaginary axis are in danger of being inundated with the collapsing modified spurious DGTD modes.

In Table 2 we show the spectral radius for each β value tried. It is clear that this simple modification yields significant reduction in the spectral radius, and corresponding increase in the maximum size of dt.

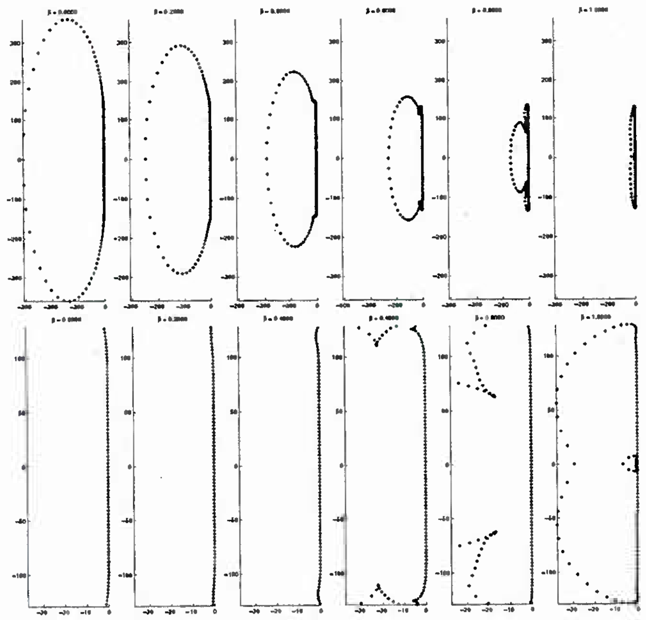


Figure 7: Full spectra for $\beta = 0, 0.2, 0.4, 0.6, 0.8, 1.0$ version of the discrete upwind DGTD operator on a mesh of $K = 20$ and $N = 7$. Notice how the spectral radius diminishes in the $\beta \to 1$ limit, i.e. the fully filtered approximation of the derivative. Top: full spectra. Bottom: zoom of origin.

We may deduce that using $\beta = 0.8$ is a reasonable compromise between spectral radius deduction and maintaining separation of the spurious modes from the imaginary axis.

Table 2: Dependence of the spectral radius of the partially stabilized operator on the β parameter for the K=20, N=7 upwind DGTD operator on the $[0, 2]$ periodic interval.

β	0	0.2	0.4	0.6	0.8	1.0
Spectral radius	614.8148	495.7820	376.6496	257.2761	137.1975	129.3503
Decrease factor	1.0000	1.2401	1.6323	2.3897	4.4812	4.7531

The decrease in spectral radius is an attractive property of this approach, but it is also necessary to examine the impact of the use of the covolume meshes on the accuracy of the solution. We performed h-convergence tests for $K = 32, 64, 128, 256$, $N = 1, 2, \ldots, 9$ with $\beta = 0$ (i.e. unfiltered upwind DGTD) with $dt = h/(2(N+1))$ and then with $\beta = 1 - 2/(N+1)$ and $dt = h/5$ using an initial condition

$$u = \exp\left(-40 \sin\left((2\pi x)^2\right)\right). \tag{39}$$

We integrated in time using the $N+1$-order Runge-Kutta-Taylor method, sometimes referred to as the Jameson-Schmidt-Turkel scheme (e.g. [4]), which for a linear autonomous system is simply a recursive evaluation of the Taylor series for the matrix exponential. We estimated the rate of convergence by assuming the error has form $\epsilon = C_n h^n$ and using the Matlab function polyfit to compute a linear fit to the log of the error values: coeffs = polyfit(log([1 .5 .25 .125]),log(errors)); order = coeffs(1).

The results for the unfiltered upwind DGTD solution of the advection equation are shown in Table 3. We clearly see that for $2 \leq n \leq 7$ the order of convergence measured at time $t = 1$ is between N and $N+1$. It is clear that the $N = 1$ results are preasymptotic for this case, and the $N = 9$ and $N = 10$ cases show convergence rates diminished because the finest resolution error is near finite precision. Here we used a time step $dt = h/(2(N+1))$ which was experimentally found to be stable for all test cases, and close to the maximum dt while retaining stability.

Table 3: h-convergence results for an unfiltered upwind DGTD discretization of periodic array of pulses advecting in the periodic interval $[0,2]$ with $dt = h/(2(N+1))$. Results prone to finite precision effects are marked with $*$.

N+1	$h = 2^{-5}$	$h = 2^{-6}$	$h = 2^{-7}$	Estimated Order
2	5.636576e-01	3.506396e-01	1.462750e-01	0.973068
3	2.226571e-01	4.608446e-02	4.562249e-03	2.804467
4	4.449859e-02	2.294515e-03	1.286897e-04	4.216860
5	9.280798e-03	2.022760e-04	6.249343e-06	5.268164
6	1.103024e-03	1.267373e-05	2.891989e-07	5.948557
7	1.096652e-04	1.625903e-06	1.288686e-08	6.527459
8	2.025857e-05	7.457799e-08	4.831807e-10	7.677805
9	2.265482e-06	1.065869e-08	1.956979e-11*	8.410415*
10	5.625579e-07	2.223321e-10	2.431499e-12*	8.909898*

Results with filtering and a fixed time step of size $dt = h/5$ are shown in Table 4. It is clear that the near optimal rates of convergence are attained in this case.

Table 4: Filtered h-convergence results for a periodic array of pulses advecting in the periodic interval $[0,2]$ with $dt = h/5$. Results prone to finite precision effects are marked with $*$.

N+1	$h = 2^{-5}$	$h = 2^{-6}$	$h = 2^{-7}$	Estimated Order
2	5.526314e-01	3.321274e-01	1.317145e-01	1.034451
3	2.269038e-01	5.524013e-02	6.671504e-03	2.543963
4	5.991306e-02	3.797743e-03	2.005506e-04	4.111380
5	5.076105e-03	1.905365e-04	7.344839e-06	4.716388
6	7.340714e-04	1.554913e-05	4.416766e-07	5.349357
7	2.419079e-04	2.224695e-06	1.488670e-08	6.994073
8	3.333436e-05	8.429268e-08	4.038341e-10	8.166444
9	4.969031e-06	7.334270e-09	3.320344e-11*	8.595636*
10	5.725566e-07	3.664249e-10	1.267431e-12*	9.392576*

In Figure 8 we show the discrete eigenspectra of the discrete DG spatial operator scaled by $dt = h/5$, this more conservative time step comfortably accommodates the discrete spectra within the RK stability regions. Some weak growth in the spectral radius of the scaled operators may be associated with the β weighted sum of filtered and unfiltered gradients.

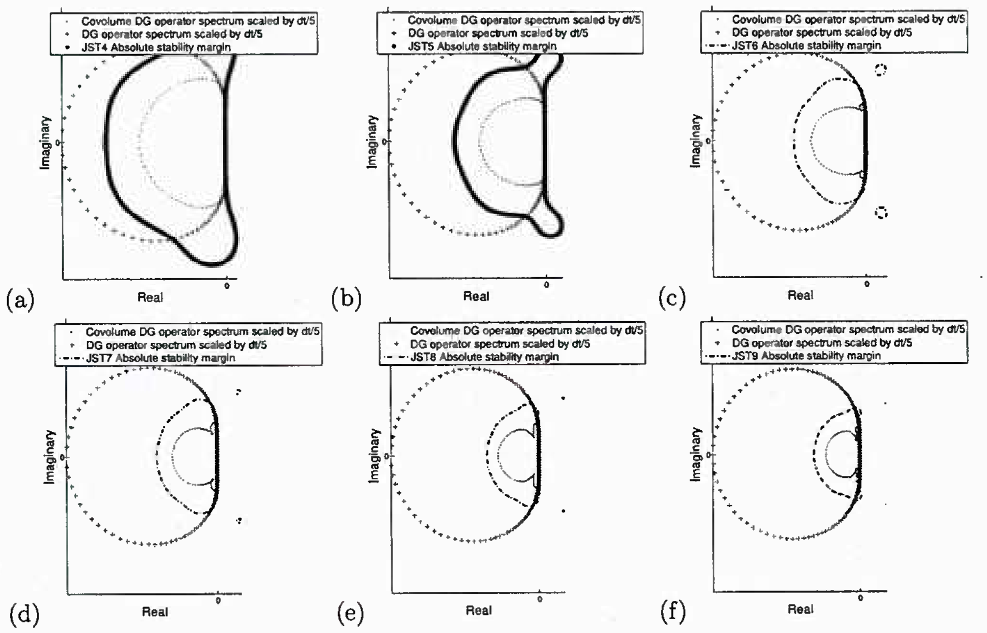


Figure 8: Discrete eigenvalues of upwind DGTD scaled by $dt = h/5$ with and without covolume mesh filtering, superimposed on the margins of absolute stability of equal order Runge-Kutta-Taylor schemes. (a) N=3, (b) N=4, (c) N=5, (d) N=6, (e) N=7, (f) N=8.

4.4 Numerical Results: 1D Maxwell's Equation

In this section we will present experimental results for the accuracy of the modified DGTD method for the 1D Maxwell's equations.

We first present estimates of the order of accuracy of the method based on simulations run to time $t = 100.23$. The model solution was constructed from a random coefficient linear combination of the first two hundred standing wave solutions to the Maxwell's equations which satisfy the boundary conditions

$$H_y = \sum_{m=1}^{m=200} r_m \cos(m\pi x)\cos(m\pi t) \ , \tag{40}$$

$$E_z = -\sum_{m=1}^{m=200} r_m \sin(m\pi x)\sin(m\pi t) \ , \tag{41}$$

where the r_m are random numbers generated by Matlab's rand function as $r_m = \text{rand}(1) * 2^{-.2m}$ which decay with with increasing mode number to guarantee smoothness. We deliberately chose a relatively long integration time to allow for the truncation error to accumulate to a measurable degree. The time step size dt was chosen according to

$$dt = 0.15h \ .$$

The results in Table 5 show that for $N \geq 4$ we see expected order of accuracy. The $N = 1$ and $N = 2$ results are likely preasymptotic and no conclusions are drawn. It should be noted that two different ranges of h were used to estimate the order of the method for $1 \leq N \leq 6$ and $7 \leq N \leq 9$ since the solutions converged to machine precision for the latter range of N before the end of the range of h used for the former tests.

Finally, we note that the $N = 1$ case with our choice of $\beta = 1 - 2/(N+1)$ is actually the unmodified upwind DGTD method.

Table 5: Order estimates for error in solving 1D Maxwell's to time $t = 100.23$ on a sequence of h-refined meshes with $1 \leq n \leq 9$. Results which are suspected to be pre-asymptotic are marked with $*$.

$(N+1)$	$h = 0.03125$	$h = 0.015625$	$h = 0.0078125$	Est. order
2	0.825*	0.479*	0.21*	1.0
3	0.158*	0.0809	0.0237	1.4
4	0.0669	0.00823	0.000595	3.4
5	0.0131	0.000542	1.5e-05	4.9
6	0.00123	2.39e-05	4.1e-07	5.8
$(N+1)$	$h = 0.0625$	$h = 0.03125$	$h = 0.015625$	Est. order
7	0.0125	0.000293	1.69e-06	6.4
8	0.00282	3.35e-05	7.56e-08	7.6
9	0.000937	4.67e-06	3.66e-09	9.0
10	0.000433	9.52e-07	6.2e-10	9.7

4.5 Summary: Filtering Relaxes Time Step Restriction

During this project period we have created a simple modification to the upwind DGTD methods which increases the allowable time step by a factor proportional to the polynomial order of the spatial discretization, N. Both a basic local truncation error type analysis and Fourier based dispersion analysis reveals that the modified method yields convergence rates comparable to the unmodified method in L^2. The method was tested for the advection equation and Maxwell's equations on one-dimensional meshes. We have conducted similar experiments on two-dimensional meshes. Unfortunately the addition of multi-diemnsional co-volume meshes presents logistical difficulties for the DGTD method. Nontheless this is the first demonstration of a filter based approach that achieves the CFL modification without impacting the solution accuracy. Further work is required to create a practical version.

5 Complete Radiation Boundary Conditions

A central issue in the development of efficient computaional methods for simulating time-domain wave propagation is the imposition of accurate, near-field radiation boundary conditions. In recent years a number of new techniques have been proposed which are capable of providing arbitrary accuracy [70,71,75]. Foremost among these fast, low-memory algorithms for evaluating integral operators appearing in exact formulations [56,57,82], the perfectly matched layer (PML) [58,61], and arbitrary-order local radiation boundary condition sequences [68, 69, 72, 77]. The integral equation formulations have the advantage of excellent long-time accuracy, but require special boundary shapes and, in $3 + 1$ dimensions, spherical harmonic transforms. The local methods, on the other hand, can be used on polygonal artificial boundaries, but become costly if long-time accuracy is required [63–65].

The goal of this aspect of the project is to develop a method which combines the excellent long-time accuracy of the nonlocal conditions with the geometric flexibility of the local methods. Following the constructions in [68, 69, 72, 77], we focus on auxiliary variable formulations applicable on polygonal artificial boundaries. Precisely we prove that the number of auxiliary variables, P, required to obtain an accuracy ϵ

up to time T satisfies:

$$P = O\left(\ln\frac{1}{\epsilon}\cdot\ln\frac{cT}{\delta}\right) \tag{42}$$

where $\delta > 0$ is the distance from the artificial boundary to the domain containing sources, scatterers, or other inhomogeneities. For efficient discretization it is natural to choose $\delta \sim \lambda$ where λ is the smallest wavelength of interest. Then the asymptotic complexity matches that of the optimized spatially nonlocal approximations constructed in [57]. Thus, in our view, the proposed method provides a satisfactory solution to the time-domain domain truncation problem for isotropic systems, and it has the potential for generalization to more complex models.

We also note that, although our focus here is on auxiliary variable formulations of local boundary condition sequences, it is now known that such sequences can be viewed as specific discretizations of perfectly matched layers [59,69]. Thus our construction also suggests a certain combination of real and complex grid stretching to optimize PML discretizations. This will be pursued in future work.

5.1 Complete plane wave representations

We begin by considering a field, $u(x,y,t)$, which satisfies the scalar wave equation in the half-space $x > -\delta$ for some $\delta > 0$. That is:

$$\frac{\partial^2 u}{\partial t^2} = c^2\nabla^2 u, \quad x > -\delta, \quad y \in \mathbb{R}^{d-1}, \quad t > 0, \tag{43}$$

with $u = 0$ at $t = 0$. Here we suppose the field is produced by sources, scatterers, and other inhomogeneities located in the half-space $x < -\delta$. These effects are all accounted for by Dirichlet data:

$$u(-\delta, y, t) = g(y,t). \tag{44}$$

We then easily derive a formula for the evolution in x of the transverse-Fourier-Laplace transforms of the solution of (43)-(44):

$$\hat{u}(x,k,s) = \hat{u}(-\delta,k,s)e^{-(\bar{s}^2+|k|^2)^{1/2}(x+\delta)}. \tag{45}$$

Here $k \in \mathbb{R}^{d-1}$ are the dual Fourier variables to the transverse spatial coordinates y, $|k|$ is the Euclidean norm, s is the dual Laplace variable to time, and $\bar{s} = \frac{s}{c}$. The branch of the square root is chosen to have positive real part for $\mathbb{R}s > 0$.

Now invert the transform, integrating with respect to k over the real hyperplane $\mathbb{R}^{d-1}$ and with respect to s over the contour, $s = i\omega + \frac{1}{T}$ with ω real. Here $T > 0$ is fixed and has units of time. For numerical methods T will measure the time over which approximations based on the exact representation are valid. We begin by representing the square root. Setting

$$k = \frac{\tilde{k}}{cT}, \quad \omega = \frac{\tilde{\omega}}{T}, \tag{46}$$

we have

$$\left((1+i\tilde{\omega})^2 + \tilde{k}^2\right)^{1/2} = a + ib, \tag{47}$$

where by squaring we deduce

$$b = \frac{\tilde{\omega}}{a}, \quad 1 + \tilde{k}^2 - \tilde{\omega}^2 = a^2 - \frac{\tilde{\omega}^2}{a^2}. \tag{48}$$

As the right-hand side of the second equation above is an increasing function of a^2 and is not greater than the left-hand side when $a^2 = 1$ we conclude that for the branch of the square root chosen $a \geq 1$. Thus we have for some $\phi \in [0, \frac{\pi}{2})$:

$$a = \frac{1}{\cos\phi}, \quad b = \tilde{\omega}\cos\phi. \tag{49}$$

Reintroducing s we finally have:

$$(\bar{s}^2 + |k|^2)^{1/2} = \bar{s}\cos\phi + \frac{1}{cT}\frac{\sin^2\phi}{\cos\phi}, \tag{50}$$

where

$$s = \frac{1}{T} + i\omega, \quad |k| = \frac{1}{cT}\sqrt{\tan^2\phi + \sin^2\phi\frac{\omega^2}{T^2}}. \tag{51}$$

Going over to polar coordinates, ρ, $\theta \in \mathbb{S}^{d-2}$, in the dual spatial variables and replacing $\rho = |k|$ by $\rho(\phi,\omega)$ we obtain:

$$u(x,y,t) = (2\pi)^{-\frac{(d+1)}{2}} \int_0^{\frac{\pi}{2}} \int_{-\infty}^{\infty} \int_{\mathbb{S}^{d-2}} e^{\psi}\bar{u}\rho^{d-2}\frac{d\rho}{d\phi} dA(\theta)d\omega d\phi, \tag{52}$$

where

$$\psi = \left(i\omega + \frac{1}{T}\right)\left(t - \frac{\cos\phi}{c}(x+\delta)\right) + i\rho(\theta\cdot y) - \frac{1}{cT}\cdot\frac{\sin^2\phi}{\cos\phi}(x+\delta), \tag{53}$$

$$\bar{u} = \hat{u}\left(-\delta, \rho\theta, \frac{1}{T} + i\omega\right). \tag{54}$$

Setting

$$\Phi(t,y,\phi) = (2\pi)^{-\frac{(d+1)}{2}} \int_{-\infty}^{\infty} \int_{\mathbb{S}^{d-2}} e^{(i\omega + \frac{1}{T})t + i\rho(\theta\cdot y)}\bar{u}\rho^{d-2}\frac{d\rho}{d\phi} dA(\theta)d\omega \tag{55}$$

we finally have our complete plane wave representation of the solution, valid for $x > -\delta$:

$$u(x,y,t) = \int_0^{\frac{\pi}{2}} \Phi\left(t - \frac{\cos\phi}{c}(x+\delta), y, \phi\right) e^{-\frac{1}{cT}\cdot\frac{\sin^2\phi}{\cos\phi}(x+\delta)} d\phi. \tag{56}$$

Note that an alternative representation can be derived by inverting the Laplace transform along the imaginary axis. Then we express the solution as a superposition of propagating plane waves at all possible angles and evanescent waves at all possible decay rates. Such an expression has been analyzed in the time domain by Heyman [79] and plays an important role in the plane wave fast time-domain algorithm [67]. We have also used it to derive radiation boundary conditions [73]. The expression given here is somewhat more efficient for first order systems, but the alternative expression given in [73] has some advantages for second order formulations.

We are, of course, aware that calling this a plane wave representation is a misnomer due to the y-dependence of Φ, but we use the term for simplicity.

5.1.1 Extension to systems and other generalizations

To extend (56) to isotropic systems we simply note that the representation above holds for any field component which satisfies the scalar wave equation. For example, suppose the equations of acoustics hold in $x > -\delta$:

$$\frac{\partial p}{\partial t} + c\nabla\cdot v = 0, \tag{57}$$

$$\frac{\partial v}{\partial t} + c\nabla p = 0. \tag{58}$$

Then by our assumptions on the initial data $\nabla \times v = 0$ so we may introduce a velocity potential, $v = \nabla q$, $cp = -\frac{\partial q}{\partial t}$. Then q satsfies the scalar wave equation and thus p and v admit the plane wave representation (56). Similar considerations hold for Maxwell's equations.

The representation may also apply to systems with multiple wave speeds. Consider, for example, Navier's equations of linear elasticity:

$$\rho \frac{\partial^2 w}{\partial t^2} = (\lambda + \mu)\nabla\nabla \cdot w + \mu \nabla^2 w. \tag{59}$$

Introducing the Helmholtz decomposition of w:

$$w = \nabla q + \nabla \times \eta, \quad \nabla \cdot \eta = 0, \tag{60}$$

we derive scalar wave equations for q and the components of η:

$$\frac{\partial^2 q}{\partial t^2} = c_1^2 \nabla^2 q, \quad \frac{\partial^2 \eta}{\partial t^2} = c_2^2 \nabla^2 \eta, \tag{61}$$

where

$$c_1^2 = \frac{\lambda + 2\mu}{\rho}, \quad c_2^2 = \frac{\mu}{\rho}. \tag{62}$$

Thus q and η satsify (56), albeit with different wavespeeds, and hence w satsfies a combined representation:

$$\begin{aligned} w_i(x, y, t) \quad = \quad & \int_0^{\frac{\pi}{2}} \Phi_{1,i} \left(t - \frac{\cos\phi}{c_1}(x + \delta), y, \phi \right) e^{-\frac{1}{c_1 T} \cdot \frac{\sin^2 \phi}{\cos \phi}(x+\delta)} d\phi \\ & + \int_0^{\frac{\pi}{2}} \Phi_{2,i} \left(t - \frac{\cos\phi}{c_2}(x + \delta), y, \phi \right) \mathrm{e}^{-\frac{1}{c_2 T} \cdot \frac{\sin^2 \phi}{\cos \phi}(x+\delta)} d\phi. \end{aligned} \tag{63}$$

Lastly we consider an anisotropic example, the acoustics system in a subsonic mean flow:

$$\frac{Dp}{Dt} + c\nabla \cdot v = 0, \tag{64}$$

$$\frac{Dv}{Dt} + c\nabla p = 0, \tag{65}$$

where

$$\frac{D}{Dt} \equiv \frac{\partial}{\partial t} + V \cdot \nabla, \quad |V| < c. \tag{66}$$

Now we introduce the Helmholtz decomposition of v:

$$w = \nabla q + \nabla \times \eta, \quad cp = -\frac{Dq}{Dt}, \tag{67}$$

leading to the equations:

$$\frac{D^2 q}{Dt^2} = c^2 \nabla^2 q, \tag{68}$$

$$\frac{D\eta}{Dt} = 0. \tag{69}$$

The solution of the transport equation (69) in the half-space is dependent on the sign of V_x. If $V_x \leq 0$ then $\eta = 0$ but if $V_x > 0$ then:

$$\eta = \Sigma(V_x t - (x + \delta), V_y t - y). \tag{70}$$

As for the convective wave equation (68) the Fourier-Laplace transform of the solution of the Dirichlet problem analogous to (45) is:

$$\hat{q}(x, k, s) = \hat{q}(-\delta, k, s) \mathrm{e}^{\frac{\bar{s}V_x - c(\bar{s}^2 + (c^2 - V_x^2)|k|^2)^{1/2}}{c^2 - V_x^2}(x+\delta)}, \tag{71}$$

with

$$\tilde{s} = s + iV_y \cdot k. \tag{72}$$

Following the same decomposition of the inversion integrals as in the case of the standard wave equation yields the slightly more complex formula:

$$q(x,y,t) = \int_0^{\frac{\pi}{2}} \Phi\left(t - \frac{\mathrm{c}\cos\phi - V_x}{c^2 - V_x^2}(x+\delta), y - V_y t, \phi\right) e^{-\frac{\mathrm{c}}{(\mathrm{c}^2 - V_x^2)T} \cdot \frac{\sin^2\phi}{\cos\phi}(x+\delta)} d\phi. \tag{73}$$

The formula clearly displays the possibility of incoming phase velocities for outgoing waves at outflow. We have not as yet considered the problem of constructing stable boundary conditions based on this representation, but we plan to do so in the future. This issue is closely related to the problem of constructing stable perfectly matched layers for anisotropic systems, which is still not completely understood [58, 60].

5.2 Approximate local boundary condition sequences

We now use (56) to derive approximate local boundary condition sequences. We focus on first order systems; the treatment of second order formulations is discussed in [76].

By way of motivation, consider an approximation to (56) derived by replacing the ϕ integral by a quadrature rule with nodes ϕ_j and weights h_j:

$$u(x,y,t) \approx \sum_{j=0}^{P} h_j \Phi\left(t - \frac{\cos\phi_j}{\mathrm{c}}(x+\delta), y, \phi_j\right) e^{-\frac{1}{cT} \cdot \frac{\sin^2\phi_j}{\cos\phi_j}(x+\delta)}. \tag{74}$$

Generalizing the construction of [77], we introduce a second set of angles $\bar{\phi}_j$ and define auxiliary functions $u_j(x,y,t)$ by setting $u_0 = u$ and recursively solving in $x > -\delta$:

$$\frac{\cos\bar{\phi}_j}{\mathrm{c}}\frac{\partial u_{j+1}}{\partial t} - \frac{\partial u_{j+1}}{\partial x} + \frac{1}{cT}\frac{\sin^2\bar{\phi}_j}{\cos\bar{\phi}_j}u_{j+1} = \frac{\cos\phi_j}{\mathrm{c}}\frac{\partial u_j}{\partial t} + \frac{\partial u_j}{\partial x} + \frac{1}{cT}\frac{\sin^2\phi_j}{\cos\phi_j}u_j, \tag{75}$$

subject to $u_{j+1}(x,y,0) = 0$. It is straightforward to see that the individual terms in (74) are annihilated by one of the differential operators on the right-hand side of (75). Thus if we replaced u_0 by the approximate representation we would conclude:

$$u_{P+1} = 0. \tag{76}$$

We now simply impose (76) on incoming normal characteristic variables. Precisely, consider the first order hyperbolic system:

$$\frac{\partial w}{\partial t} + A\frac{\partial w}{\partial x} + \sum_{j=1}^{d-1} B_j \frac{\partial w}{\partial y_j} = 0, \tag{77}$$

where we have put A in diagonal form:

$$A = \begin{pmatrix} D_+ & 0 & 0 \\ 0 & -D_- & 0 \\ 0 & 0 & 0 \end{pmatrix}, \tag{78}$$

and the diagonal matrices $D_\pm$ are positive. Block w according to the blocks of A:

$$w = \begin{pmatrix} w_+ \\ w_- \\ w_c \end{pmatrix}. \tag{79}$$

As in [77], a simple induction argument shows that vector functions w_j defined via (75) (with u replaced by w) will also satisfy (78). Multiplying the recursion relations by A and eliminating x derivatives yields the following system which can be considered along the boundary $x = 0$:

$$\left(I + \frac{\cos\bar{\phi}_j}{c} A\right) \frac{\partial w_{j+1}}{\partial t} + \sum_{k=1}^{d-1} B_k \frac{\partial w_{j+1}}{\partial y_k} + \frac{1}{cT} \frac{\sin^2\bar{\phi}_j}{\cos\bar{\phi}_j} A w_{j+1} = \left(-I + \frac{\cos\phi_j}{c} A\right) \frac{\partial w_j}{\partial t} - \sum_{k=1}^{d-1} B_k \frac{\partial w_j}{\partial y_k} + \frac{1}{cT} \frac{\sin^2\phi_j}{\cos\phi_j} A w_j. \tag{80}$$

Imposing the termination condition:

$$w_{P+1,-} = 0, \tag{81}$$

we see that (80) implicitly defines a relationship between the outgoing characteristic variables $w_{0,+}$ and the incoming characteristic variables, $w_{0,-}$. Precisely, if we assume the functions w_j are given along the boundary $x = 0$ for some time t and in addition that $\frac{\partial w_{0,+}}{\partial t}$ is known, then the remaining time derivatives may be directly computed. For the auxiliary variables associated with the outgoing characteristics we may solve for increasing j:

$$\begin{aligned}\left(I + \frac{\cos\bar{\phi}_j}{c} D_+\right) \frac{\partial w_{j+1,+}}{\partial t} = {} & \left(-I + \frac{\cos\phi_j}{c} D_+\right) \frac{\partial w_{j,+}}{\partial t} \\ & - \left(\sum_{k=1}^{d-1} B_k \left(\frac{\partial w_{j+1}}{\partial y_k} \frac{\partial w_j}{\partial y_k}\right)\right)_+ \\ & + \frac{1}{cT} D_+ \left(\frac{\sin^2\phi_j}{\cos\phi_j} w_{j,+} - \frac{\sin^2\bar{\phi}_j}{\cos\bar{\phi}_j} w_{j+1,+}\right).\end{aligned} \tag{82}$$

For the auxiliary variables associated with the incoming variables, on the other hand, we solve for decreasing j:

$$\begin{aligned}\left(I + \frac{\cos\phi_j}{c} D_-\right) \frac{\partial w_{j,-}}{\partial t} = {} & \left(-I + \frac{\cos\bar{\phi}_j}{c} D_-\right) \frac{\partial w_{j+1,-}}{\partial t} \\ & - \left(\sum_{k=1}^{d-1} B_k \left(\frac{\partial w_{j+1}}{\partial y_k} \frac{\partial w_j}{\partial y_k}\right)\right)_- \\ & - \frac{1}{cT} D_- \left(\frac{\sin^2\phi_j}{\cos\phi_j} w_{j,-} - \frac{\sin^2\bar{\phi}_j}{\cos\bar{\phi}_j} w_{j+1,-}\right).\end{aligned} \tag{83}$$

Lastly the time derivatives of the $w_{j,c}$ may be computed using the hyperbolic system itself:

$$\frac{\partial w_{j,c}}{\partial t} = -\left(\sum_{k=1}^{d-1} B_k \frac{\partial w_j}{\partial y_k}\right)_c. \tag{84}$$

We note that the implementation of this method within a standard time-stepping procedure is straightforward. We will discuss below how this has been done for difference methods, while in [87] discontinuous Galerkin methods are used. As presented so far, the formulation may seem somewhat ad hoc. However, it will be shown in the next section that it is equivalent to a rational interpolant of the symbol of nonlocal operators arising in exact boundary condition representations, with the interpolation nodes determined by both ϕ_j and $\bar{\phi}_j$.

5.3 Analysis

To derive error estimates we must estimate the complex reflection coefficient. For simplicity we will carry out the analysis for the acoustic system with $d = 3$, but similar estimates for Maxwell's equations and for other dimensions could be derived in the same way. (See [73] for a direct treatment of the scalar wave equation.)

We consider the system in diagonalized form and slightly change notation:

$$\frac{\partial w_+}{\partial t} + c\frac{\partial w_+}{\partial x} + c\nabla_{\tan} \cdot v_{\tan} = 0, \tag{85}$$

$$\frac{\partial w_-}{\partial t} - c\frac{\partial w_-}{\partial x} + c\nabla_{\tan} \cdot v_{\tan} = 0, \tag{86}$$

$$\frac{\partial v_{\tan}}{\partial t} + \frac{c}{2}\nabla_{\tan}(w_+ + w_-) = 0, \tag{87}$$

where

$$w_+ = p + v_x, \quad w_- = p - v_x. \tag{88}$$

Specialized to this sytem equations (82)-(84) become:

$$\begin{aligned} \frac{1+\cos\bar{\phi}_j}{c}\frac{\partial w_{+,j+1}}{\partial t} &= \frac{-1+\cos\phi_j}{c}\frac{\partial w_{+,j}}{\partial t} - \nabla_{\tan}\cdot(v_{\tan,j+1} + v_{\tan,j}) \\ &+ \frac{1}{cT}\left(\frac{\sin^2\phi_j}{\cos\phi_j}w_{+,j} - \frac{\sin^2\bar{\phi}_j}{\cos\bar{\phi}_j}w_{+,j+1}\right), \end{aligned} \tag{89}$$

$$\begin{aligned} \frac{1+\cos\phi_j}{c}\frac{\partial w_{-,j}}{\partial t} &= \frac{-1+\cos\bar{\phi}_j}{c}\frac{\partial w_{-,j+1}}{\partial t} - \nabla_{\tan}\cdot(v_{\tan,j+1} + v_{\tan,j}) \\ &+ \frac{1}{cT}\left(\frac{\sin^2\bar{\phi}_j}{\cos\bar{\phi}_j}w_{-,j+1} - \frac{\sin^2\phi_j}{\cos\phi_j}w_{-,j}\right), \end{aligned} \tag{90}$$

$$\frac{\partial v_{\tan,j}}{\partial t} + \frac{c}{2}\nabla_{\tan}(w_{+,j} + w_{-,j}) = 0. \tag{91}$$

Assume that the solution is produced by sources and Cauchy data in the region $x < -\delta$ and that the artificial boundary is located at $x = 0$. The error then satisfies the homogeneous system with zero initial data driven only by reflections from the artificial boundary. These we estimate using the data at $x = -\delta$, which is crucial if positive results are to be obtained. (Roughly, the derivation of finite time estimates with separation between sources and the boundary excludes glancing modes.)

Performing a Fourier-Laplace transformation as before, the solution for $x > -\delta$ can be completely characterized by $\hat{w}_+(-\delta, k, s)$:

$$\begin{pmatrix} \hat{w}_+ \\ \hat{w}_- \\ \hat{v}_{\tan} \end{pmatrix} = \hat{w}_+(-\delta, k, s)e^{-\gamma(x+\delta)} \begin{pmatrix} 1 \\ \frac{\bar{s}-\gamma}{\bar{s}+\gamma} \\ \frac{-ik}{\bar{s}+\gamma} \end{pmatrix}, \tag{92}$$

where we have introduced

$$\gamma = \left(\bar{s}^2 + |k|^2\right)^{\frac{1}{2}}. \tag{93}$$

Similarly, the error is a reflected wave completely characterized by $\hat{e}_-(0, k, s)$:

$$\begin{pmatrix} \hat{e}_+ \\ \hat{e}_- \\ \hat{e}_{\tan} \end{pmatrix} = \hat{e}_-(0, k, s)e^{\gamma x} \begin{pmatrix} \frac{\bar{s}-\gamma}{\bar{s}+\gamma} \\ 1 \\ \frac{-ik}{\bar{s}+\gamma} \end{pmatrix}. \tag{94}$$

Now $\hat{e}_-(0, k, s)$ can be directly calculated using the boundary recursion.

Lemma 5.1 *The reflection from the artificial boundary is given by:*

$$\hat{e}_-(0,k,s) = R(k,s)\hat{w}_+(-\delta,k,s), \tag{95}$$

where

$$\begin{aligned} R(k,s) \quad &= \quad \left(\prod_{j=0}^{P}\left(\frac{\gamma - \bar{s}\cos\phi_j - \frac{1}{cT}\frac{\sin^2\phi_j}{\cos\phi_j}}{\gamma + \bar{s}\cos\phi_j + \frac{1}{cT}\frac{\sin^2\phi_j}{\cos\phi_j}}\right)\cdot\left(\frac{\gamma - \bar{s}\cos\bar{\phi}_j - \frac{1}{cT}\frac{\sin^2\bar{\phi}_j}{\cos\phi_j}}{\gamma + \bar{s}\cos\bar{\phi}_j + \frac{1}{cT}\frac{\sin^2\bar{\phi}_j}{\cos\bar{\phi}_j}}\right)\right) \\ &\quad \times\left(\frac{\gamma-\bar{s}}{\gamma+\bar{s}}\right)e^{-\gamma\delta}. \end{aligned} \tag{96}$$

Using this explicit representation for the reflection coefficient we can directly establish the well-posedness of the system with the approximate boundary and write down error estimates.

Theorem 5.2 *The initial-boundary value problem for the acoustic system in the half-space $x < 0$ with the complete radiation boundary conditions (80)-(81) is strongly well-posed. Moreover there exists a universal constant C such that for $x < 0$:*

$$\|e(x,\cdot,\cdot)\|_{L_2(\mathbb{R}^{d-1}\times(0,T))} \leq \rho C\|w(-\delta,\cdot,\cdot)\|_{L_2(\mathbb{R}^{d-1}\times(0,T))}, \tag{97}$$

where

$$\rho = \max_{0\leq\phi\leq\frac{\pi}{2}}\left(\prod_{j=0}^{P}\frac{|\cos\phi - \cos\phi_j|}{(\cos\phi+\cos\phi_j)}\frac{|\cos\phi-\cos\bar{\phi}_j|}{(\cos\phi+\cos\bar{\phi}_j)}\right)\cdot\left(\frac{1-\cos\phi}{1+\cos\phi}\right)e^{-\frac{\delta}{cT}\cdot\frac{1}{\cos\phi}}. \tag{98}$$

5.4 Parameter selection

According to (98), the error estimate is optimized by choosing angles, ϕ_j, to minimize ρ. Here we adapt Newman's well-known construction of rational approximants to $|x|$ [83] to derive an explicit set of angles which achieve the tolerance with $P = O(\ln\frac{1}{\epsilon}\cdot\ln\frac{cT}{\delta}$. (See [84] for a further discussion of optimal approximations to similar functions.)

Choosing a tolerance, $\epsilon > 0$, we note that both the exponential factor and the parameter-dependent factor in (98) are bounded above by one. Thus we can satisfy the tolerance by making either less than ϵ. We begin by considering the regime where $\cos\phi$ is small. In particular we have:

$$e^{-\frac{\delta}{cT}\cdot\frac{1}{\cos\phi}} < \epsilon, \tag{99}$$

whenever

$$\cos\phi < \frac{\delta}{cT}\frac{1}{\ln\frac{1}{\epsilon}} \equiv c_0. \tag{100}$$

Now restrict attention to the interval $c_0 \leq \cos\phi \leq 1$ and consider the problem of minimizing:

$$\rho_c \equiv \max_{c_0\leq c\leq 1}\left(\prod_{j=0}^{2P+1}\frac{|c-a_j|}{c+a_j}\right)\cdot\left(\frac{1-c}{1+c}\right), \tag{101}$$

where now the parameters a_j encompass both $\cos\phi_j$ and $\cos\bar{\phi}_j$. Following [83] set:

$$a_j = g^{j+1}, \quad g = (c_0)^{\frac{1}{2P+2}}. \tag{102}$$

Suppose

$$g^{j+1} < c < g^j. \tag{103}$$

Then after some manipulation

$$\rho_c \leq \prod_{k=1}^{2P+3} \exp\left(-2g^k\right) = \exp\left(-2g(1-g^{2P+2})/(1-g)\right).$$

We thus must choose P sufficiently large that:

$$2g\frac{1-g^{2P+2}}{1-g} > \ln\frac{1}{\epsilon}. \tag{104}$$

Assuming for simplicity that $\epsilon \ll 1$, $\frac{cT}{\delta} \gg 1$ it is sufficient to satisfy:

$$g > 1 - \frac{1}{\ln\frac{1}{\epsilon}} \tag{105}$$

which requires

$$P = O\left(\ln\frac{1}{c_0} \cdot \ln\frac{1}{\epsilon}\right). \tag{106}$$

Using Theorem 5.2 we have thus proven:

Theorem 5.3 *Suppose* $\epsilon < 1$ *and* $\frac{cT}{\delta} > \ln\frac{1}{\epsilon}$. *Then there exists a universal constant* C *such that if:*

$$P > C\ln\frac{cT}{\delta} \cdot \ln\frac{1}{\epsilon}, \tag{107}$$

and the complete radiation boundary conditions (80)-(81) are imposed with parameters:

$$\cos\phi_j = c_0^{\frac{2j+1}{2P+2}}, \quad \cos\bar{\phi}_j = c_0^{\frac{j+1}{P+1}}, \tag{108}$$

for $j = 0, \dots, P$ *and* c_0 *defined by (100) the error satisfies:*

$$\|e(x,\cdot,\cdot)\|_{L_2(\mathbb{R}^{d-1}\times(0,T))} \leq \epsilon\|w(-\delta,\cdot,\cdot)\|_{L_2(\mathbb{R}^{d-1}\times(0,T))}. \tag{109}$$

Clearly this analysis, by considering only one of the factors in (98) at a time, does not produce optimal error estimates or optimal parameters. For practical purposes we will compute parameters numerically by minimizing ρ for fixed values of $\frac{\delta}{cT}$ and P. We employ the standard Remez algorithm (e.g. [84]) with initial approximations provided by the geometric distribution introduced above. We note that the computation is extremely rapid, so that in practice one could directly compute a minimal P-value and the associated parameters for input values of ϵ and $\frac{\delta}{cT}$.

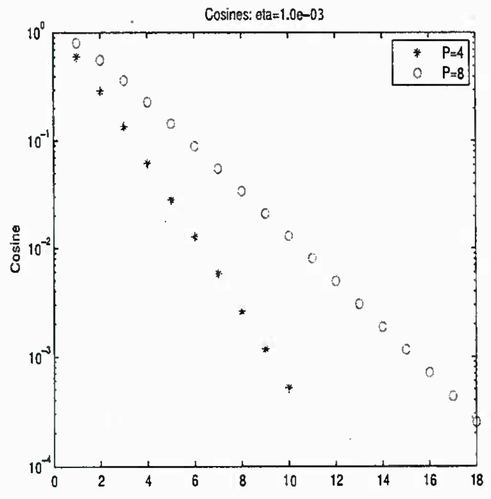
In Figure 9a we plot the optimal cosines for $\frac{\delta}{cT} = 10^{-3}$ and $P = 4, 8$ and in Figure 9bwe plot the corresponding reflection coefficients as a function of θ.

5.5 Hyperbolicity of the boundary system

Although we have proven that the initial-boundary value problem for the acoustic system combined with the approximate radiation boundary conditions is strongly well-posed, we would like to additionally consider the well-posedness of the boundary system directly. This leads to confidence that it can be directly discretized as a hyperbolic system and also motivates the form of the corner closures we will seek. To that end we consider (89)-(91) for $j = 0, \dots, P$ supplemented by (85), with $\frac{\partial w_+}{\partial x}$ treated as an inhomogeneous term and $w_{-,P+1} = 0$. In three space dimensions we then have a system of $4P+5$ first order equations on the boundary ($3P+4$ in two space dimensions). We were able to establish hyperbolicity, stated in the following Lemma:

Lemma 5.4 *The boundary system is hyperbolic with characteristic speeds less than or equal to* c.

In addition we were able to create corner closure equations..

a) 

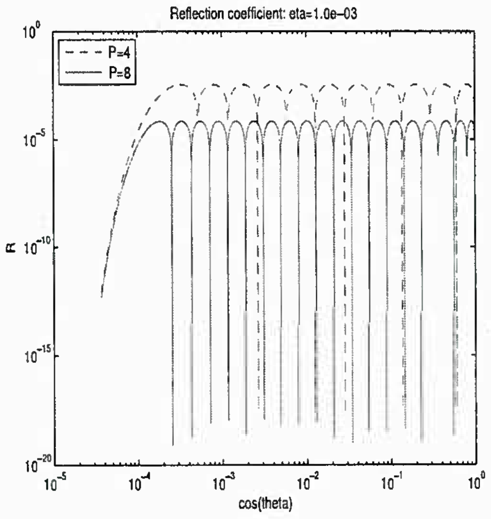
b) 

Figure 9: a) Optimal cosines for $\frac{\delta}{cT} = 10^{-3}$. b) Reflection coefficients for $\frac{\delta}{cT} = 10^{-3}$.

5.5.1 Physical boundaries

At physical boundaries the acoustic system requires a single boundary condition. We suppose it takes the general form:

$$b_p p + b_v \cdot v = g, \tag{110}$$

with $g = 0$ for $x > -\delta$. For example, we may condider the solid wall boundary condition $v \cdot n = 0$ where n is normal to the boundary. As the linear combination of variables on the left-hand side of (110) applied to the auxiliary variables p_j, v_j satisfies the basic recurison relations (75) we conclude:

$$b_p p_j + b_v \cdot v_j = 0, \quad j = 0, \ldots, P+1. \tag{111}$$

We then impose these as our boundary conditions. Note that we have not yet tried to analyze the well-posedness of the boundary system terminated with (111), but we have encountered no instabilities in numerical experiments.

5.5.2 Intersection of artificial boundaries

In this case we have no direct way to translate the boundary conditions from the adjacent boundary to termination conditions for the auxiliary variables. Instead, we attempt to generalize the closures based on compatibility conditions first proposed for homogeneous rational approximants in [62,80,86]. Our construction is closely related to the one presented in [69], and in particular it can be easily interpreted via the analogy with PML mentioned earlier as a corner layer.

To minimize the algebraic complexity, we will focus first on the case of two space dimensions. We also make the unnecessary assumptions that identical parameters and boundary condition orders are being used on each piece of the boundary and that the boundaries meet at a right angle. The more general case of boundaries meeting at angles greater than or equal to $\frac{\pi}{2}$ will be described in [87]. We identify the coordinates x and y as the normals to the two boundaries.

We now define a doubly indexed collection of auxiliary variables which satisfy both the recursions (75) associated with the two normals.

$$\frac{\cos\bar{\phi}_k}{c}\frac{\partial u_{j,k+1}}{\partial t} - \frac{\partial u_{j,k+1}}{\partial x} + \frac{1}{cT}\frac{\sin^2\bar{\phi}_k}{\cos\bar{\phi}_k}u_{j,k+1} = \frac{\cos\phi_k}{c}\frac{\partial u_{j,k}}{\partial t} + \frac{\partial u_{j,k}}{\partial x} + \frac{1}{cT}\frac{\sin^2\phi_k}{\cos\phi_k}u_{jk}, \tag{112}$$

$$\frac{\cos\bar{\phi}_j}{c}\frac{\partial u_{j+1,k}}{\partial t} - \frac{\partial u_{j+1,k}}{\partial y} + \frac{1}{cT}\frac{\sin^2\bar{\phi}_j}{\cos\bar{\phi}_j}u_{j+1,k} = \frac{\cos\phi_j}{c}\frac{\partial u_{j,k}}{\partial t} + \frac{\partial u_{j,k}}{\partial y} + \frac{1}{cT}\frac{\sin^2\phi_j}{\cos\phi_j}u_{jk}. \tag{113}$$

Note that the variables $u_{0,k}$ can be identified with the auxilary variables for the piece of the artificial boundary with normal coordinate x and $u_{j,0}$ identified with variables for the piec with normal coordinate y. Combining (112) with (113) for each component together with the acoustic system itself allows us to eliminate all spatial derivatives The only subtlety to the process comes from the fact that each direction is characteristic. The final forms we obtain are as follows. For $j,k=0,\ldots,P$:

$$\begin{aligned}
\frac{\partial p_{j+1,k+1}}{\partial t} + \frac{\partial p_{j,k+1}}{\partial t} + \frac{\partial p_{j+1,k}}{\partial t} + \frac{\partial p_{j,k}}{\partial t} \quad = \quad & \cos\phi_k\frac{\partial v_{x,j+1,k}}{\partial t} - \cos\bar{\phi}_k\frac{\partial v_{x,j+1,k+1}}{\partial t} \\
+ \quad & \cos\phi_k\frac{\partial v_{x,j,k}}{\partial t} - \cos\bar{\phi}_k\frac{\partial v_{x,j,k+1}}{\partial t} \\
+ \quad & \frac{\sin^2\phi_k}{T\cos\phi_k}v_{x,j+1,k} - \frac{\sin^2\bar{\phi}_k}{T\cos\bar{\phi}_k}v_{x,j+1,k+1} \\
+ \quad & \frac{\sin^2\phi_k}{T\cos\phi_k}v_{x,j,k} - \frac{\sin^2\bar{\phi}_k}{T\cos\bar{\phi}_k}v_{x,j,k+1} \\
+ \quad & \cos\phi_j\frac{\partial v_{y,j,k+1}}{\partial t} - \cos\bar{\phi}_j\frac{\partial v_{y,j+1,k+1}}{\partial t} \\
+ \quad & \cos\phi_j\frac{\partial v_{y,j,k}}{\partial t} - \cos\bar{\phi}_j\frac{\partial v_{y,j+1,k}}{\partial t} \\
+ \quad & \frac{\sin^2\phi_j}{T\cos\phi_j}v_{y,j,k+1} - \frac{\sin^2\bar{\phi}_j}{T\cos\bar{\phi}_j}v_{y,j+1,k+1} \\
+ \quad & \frac{\sin^2\phi_j}{T\cos\phi_j}v_{y,j,k} - \frac{\sin^2\bar{\phi}_j}{T\cos\bar{\phi}_j}v_{y,j+1,k}.
\end{aligned} \tag{114}$$

For $j=0,\ldots P+1$, $k=0,\ldots,P$:

$$\begin{aligned}
\frac{\partial v_{x,j,k+1}}{\partial t} + \frac{\partial v_{x,j,k}}{\partial t} \quad = \quad & \cos\phi_k\frac{\partial p_{j,k}}{\partial t} - \cos\bar{\phi}_k\frac{\partial p_{j,k+1}}{\partial t} \\
+ \quad & \frac{\sin^2\phi_k}{T\cos\phi_k}p_{j,k} - \frac{\sin^2\bar{\phi}_k}{T\cos\bar{\phi}_k}p_{j,k+1}.
\end{aligned} \tag{115}$$

And for $j=0,\ldots P$, $k=0,\ldots,P+1$:

$$\begin{aligned}
\frac{\partial v_{y,j+1,k}}{\partial t} + \frac{\partial v_{y,j,k}}{\partial t} \quad = \quad & \cos\phi_j\frac{\partial p_{j,k}}{\partial t} - \cos\bar{\phi}_j\frac{\partial p_{j+1,k}}{\partial t} \\
+ \quad & \frac{\sin^2\phi_j}{T\cos\phi_j}p_{j,k} - \frac{\sin^2\bar{\phi}_j}{T\cos\bar{\phi}_j}p_{j+1,k}.
\end{aligned} \tag{116}$$

Clearly we have written down $3(P+1)^2+2(P+1)$ ordinary differential equations for $3(P+2)^2$ variables. To close the system we require $4P+7$ additional equations. We obtain $2P+4$ equations by specifying "outgoing characteristic" data from each edge. Precisely, let $n_x, n_y = \pm 1$ be the components of the normal vectors to the boundaries. For $k=0,\ldots,P+1$:

$$\frac{\partial p_{0,k}}{\partial t} + n_y\frac{\partial v_{y,0,k}}{\partial y} = \mathrm{x-normal\ \ edge\ \ value}, \tag{117}$$

and for $j = 0, \ldots, P+1$:

$$\frac{\partial p_{j,0}}{\partial t} + n_x \frac{\partial v_{x,j,0}}{\partial x} = \text{y} - \text{normal edge value.} \tag{118}$$

The remaining $2P+3$ conditions follow from the termination conditions, which we average for $j = k = P+1$. For $k = 0, \ldots, P$:

$$\frac{\partial p_{P+1,k}}{\partial t} - n_y \frac{\partial v_{y,P+1,k}}{y} = 0, \tag{119}$$

for $j = 0, \ldots, P$:

$$\frac{\partial p_{j,P+1}}{\partial t} - n_x \frac{\partial v_{x,j,P+1}}{x} = 0, \tag{120}$$

and

$$2\frac{\partial p_{P+1,P+1}}{\partial t} - n_x \frac{\partial v_{x,P+1,P+1}}{x} - n_y \frac{\partial v_{y,P+1,P+1}}{y} = 0. \tag{121}$$

Solving this sytem provides us with the time derivatives of all auxiliary variables from each edge at the corner point.

In three space dimensions an analagous set of $4(P+2)^3$ equations can be written down at a corner point where three boundary faces meet. On edges we derive equations similar to the two-dimensional corner case which also include spatial derivatives along the edge. We have not yet implemented the corner closures in three space dimensions, but plan to do so in the future. We also emphasize that while the construction described above is clearly consistent with the construction of the boundary conditions and that numerical experiments show that it is stable and accurate, we have not analyzed it.

5.6 Numerical experiments

To demonstrate the effectiveness of the proposed conditions we performed two numerical experiment with the acoustic system in $2+1$ dimensions. In the first experiment we employed a 8th-order grid-stabilized difference methods in space combined with the standard 4th order Runge-Kutta method in time. (See [74] for details.) The boundary system for the auxiliary variables is approximated with exactly the same methods and grid as the interior system. The acoustic system is solved for a solution domain embedded in $\mathbb{R}^2 \times [0, 100]$ with the smooth source function:

$$s(x, y, t) = 100 \cdot (100y^2 - 1)^8 \sin^9 \pi x \sin^9 \pi t, \quad |y| \leq 0.1, \quad |x| \leq 1. \tag{122}$$

The computational domain is $(-1.1, 1.1) \times (-.2, .2)$. Now we use the complete radiation conditions on all four sides, applying the corner closure described above. Parameters are based on the choice $\frac{\delta}{cT} \equiv \eta = 10^{-3}$, $T = 100$. We record the pressure at two spatial locations: $(.2571, .071429)$,$(1.0971, .071429)$. We take $\Delta x = \Delta y = 2.8571 \times 10^{-3}$, $\Delta t = 5 \times 10^{-4}$. As we have no exact solution, errors are measured against a solution computed with a large value of P, $P = 20$. We display the results in Figure 10. Again the convergence is spectral with error levels commensurate with the reflection coefficients. In particular, we observe no adverse effects from the corner closures.

In the second experiment we performed exterior scattering from three cylindrical scatterers using a DGTD discretization in space and low storage Runge-Kutta in time. We computed a reference solution for a large domain, and then compared the solution using the complete radiation boundary conditions to truncate this into a much smaller computational domain. Our experiments again demonstrate spectral convergence with increasing numbers of auxiliary variables/equations commensurate with the reflection coefficient analysis. See Figure 11 for the truncated computational domain and convergence study.

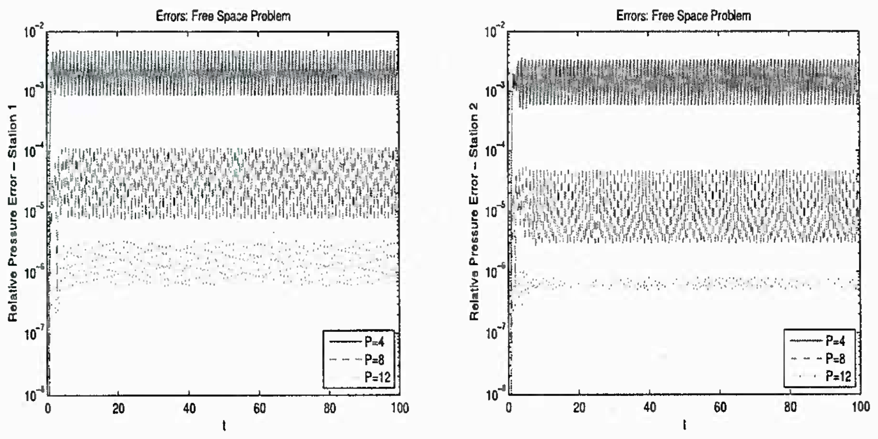


Figure 10: Pressure errors for the free space distributed source problem.

5.7 Summary: A New Family of Optimized Radiation Boundary Conditions

The new, optimized, Complete Radiation Boundary Conditions (CRBC) dramatically improves the long time behavior of existing families of high-order local radiation boundary conditions. We have optimized the integro-differential operator representation of the boundary condition to provide quasi-optimal transmission of outgoing waves including those with evanescent component.

6 Polyomino Tiling Algorithm

This section of the project report describes a very abstract topic—algorithms for generating polyomino tilings—but its motivation comes from the very concrete problem of antenna design.

In their work on the creation of wideband phased arrays of antennas, Mailloux, Santarelli, and Roberts [88] have considered working with polyomino subarrays. In the past, phased arrays would be made of identical rectangular subarrays on a grid, as in Figure 12. The periodicity of such designs results in large quantization sidelobes at certain frequencies. A *sidelobe* is a direction other than the main beam direction in which the array radiates, and sidelobes with high power relative to the main beam "represent severe pattern degradation."

The authors' cost-efficient solution was to make each subarray in the shape of a single polyomino profile. Figure 13 shows the L-octomino that they chose to use. The presence of gaps between the subarrays would reduce the aperture efficiency, so the subarrays must tile the aperture exactly, although they may extend past the edges of the aperture, as in Figure 14.

The problem that arose was the subdivision of the aperture into subarrays. They wished to create a large number of dissimilar tilings for comparison purposes, but tiling by hand proved to be tedious and time consuming. They turned to various existing polyomino software, but they found that existing programs either couldn't tile a rectangle as large as the 64×64 aperture that they used, or that they were prohibitively slow. In the end, they resorted to dividing the aperture into four 32×32 sections which could be more easily tiled.

The obvious problem with analyzing only tilings which can be divided into even quarters is that they are leaving the vast majority of tilings out of their search for ones with good properties. Furthermore, it is not unreasonable to suspect that the best tilings at avoiding the ill effects of periodicity are those which cannot

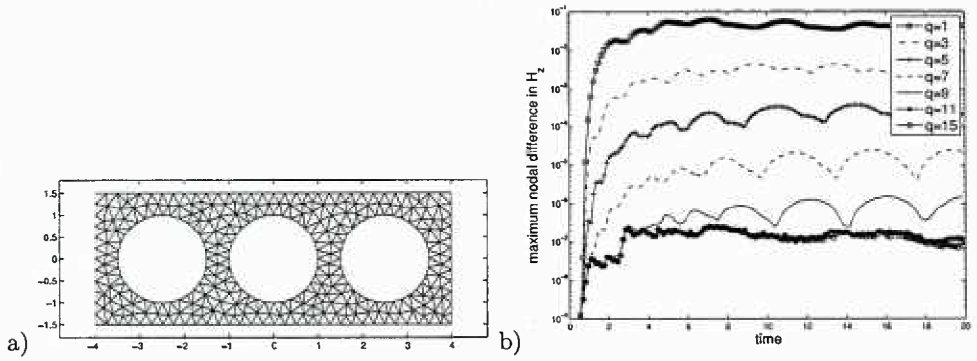


Figure 11: a): truncated domain used to compute with complete radiation boundary conditions. b): computational error for scattering in truncated domain dependence on number of auxiliary boundary equations used.

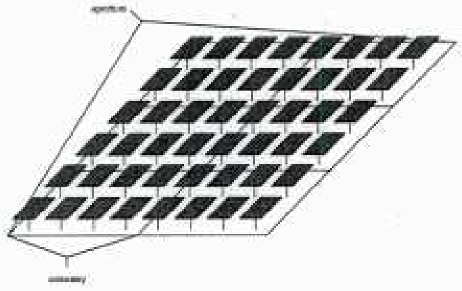


Figure 12: A diagram shower in individual antennas, a subarray, and the aperture

be cleanly divided into rectangular subsections.

We have set out to design algorithms that are better suited for the authors' purposes than the polyomino software currently in existence. In particular, we have worked with the following ideal parameters in mind:

1. The algorithm should work with an arbitrary set of polyominos, so that other subarray shapes may be explored in the future.

2. The algorithm should be reliably quick, and the time should scale well enough that larger apertures may be considered in the future (i.e. the time required to tile an $n \times n$ aperture should be $\mathcal{O}(n^p)$ for some small p).

3. The tilings generated by the program shouldn't favor any structure or pattern and should be dissimilar from each other (i.e. they should sample the space of all possible tilings somewhat uniformly).

In the background section, we will explain why traditional polyomino programs do not scale well by reviewing the relationship between polyomino tiling and the Exact Cover problem. we will review Donald Knuth's Algorithm X, the fundamental algorithm for solving the Exact Cover problem, and its behavior in different situations. We will then present three types of algorithms designed to satisfy the three criteria above. The first type builds tilings using a randomized version of Algorithm X; the second, instead of building a new tiling, modifies an existing one; the third is a hybrid of the first two. Finally, we will discuss the issue of creating a metric to score the disorder in a tiling, with the hopes that such a disorder metric will have positive correlation with high-quality phased array designs.

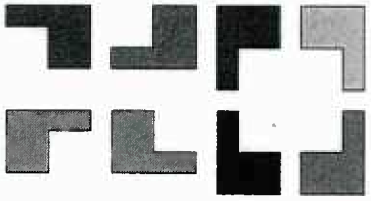


Figure 13: The L-ocotomino used by Mailloux et al., and its symmetries

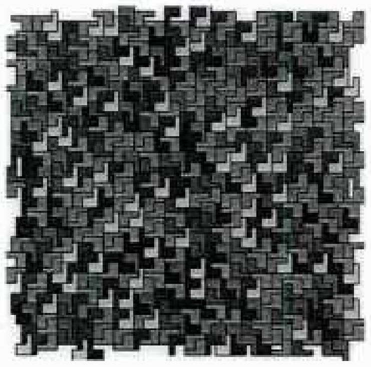


Figure 14: An example of an a tiling with subarrays extending past the edges of the aperture

6.1 Background

6.1.1 The Exact Cover Problem

The Exact Cover Problem in its most general form is to find a partition of a set within a limited collection of subsets:

> Given a set S and a collection $\mathfrak{C}$ of subsets of S, choose a subcollection $\mathfrak{C}^* \subseteq \mathfrak{C}$ such that $X \cap Y = \emptyset$ for distinct $X, Y \in \mathfrak{C}^*$ and such that $S = \bigcup_{X \in \mathfrak{C}^*} X$, or determine that no such subcollection exists.

In its discrete form, it is often helpful to envision the Exact Cover problem as a binary array. For each element $x_j \in S$ there is a row, and for each subset $X_i \in \mathfrak{C}$ there is a column. The (i, j) entry is 1 if $x_j \in X_i$ or 0 if $x_j \notin X_i$. The problem is to select a set of rows such that, within that set, a 1 appears in every column exactly once (See Figure 15).

The discrete Exact Cover problem was one of the first problems that was identified as *NP-complete*, which means that all existing algorithms for solving it are super-polynomial time algorithms in terms of the size of the inputs, and that it is very unlikely that a polynomial time algorithm exists.

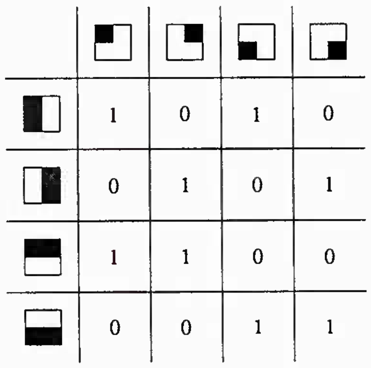
6.1.2 Describing Polyomino Tiling as an Exact Cover problem

Given a domain to be tiled with polyominos, (which need not be rectangular), S will be the set of all squares in the domain. A subset is added for each polyomino in every position where it can be legally added (see Figure 16). Thus, for a rectangular domain of size $m \times n$ that we wish to cover with polyominos from a set of k polyominos, then the size of the binary array used to visualize the Exact Cover problem will be

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
$\mathbf{X_1}$	0	**1**	**1**	0	0	0	**1**
X_2	1	0	0	1	0	0	0
$\mathbf{X_3}$	0	0	0	0	**1**	**1**	0
X_4	1	1	1	1	0	0	0
X_5	0	0	0	0	0	1	1
X_6	0	0	1	1	0	0	1
X_7	1	0	0	1	1	1	0
$\mathbf{X_8}$	**1**	0	0	**1**	0	0	0
X_9	0	1	1	1	0	0	0
X_{10}	1	0	0	1	0	0	1

Figure 15: An Exact Cover binary array, with $\{X_1, X_3, X_8\}$ as a solution

$(\sim kmn) \times mn.$ [1]



	1	0	1	0
	0	1	0	1
	1	1	0	0
	0	0	1	1

Figure 16: An example of converting a polyomino tiling problem (in this case, covering a 2×2 square with dominos) into an Exact Cover problem. Each square is assigned to a column; each placement of the vertical domino get a row, and each placement of the horizontal domino gets a row.

6.2 Algorithm X and Dancing Links

In 2000, Donald Knuth wrote down in formal terms what is essentially the trial-and-error method for solving the Exact Cover problem, calling it `Algorithm X`. He phrased it in terms of the binary array representation:

Because the row r is chosen nondeterministically, we can visualize `Algorithm X` as making a copy of itself for every r such that $A[r, c] = 1$. Thus the algorithm as stated will print *all* solutions to the given Exact Cover problem. Of course, there are no nondeterministic computers at the moment, so `Algorithm X` is of no use on its own. Thankfully, Knuth popularized a useful way of storing the binary matrix data so that two functions, `CoverColumn` and `UncoverColumn`, can be done very efficiently. `CoverColumn` takes a column c, and removes it from the binary array A, then removes every row r such that $A[r, c] = 1$ from the array; `UncoverColumn` reverses the changes made by `CoverColumn`. These functions allows us to implement `Algorithm X` deterministically. The resulting recursive algorithm, called `DLX` (the DL stand for "dancing links", the term Knuth uses for his binary matrix data structure) is as follows:

[1]Rotations and mirror-images are taken to be separate polyominos in this analysis. So the L-octomino used by Mailloux et al. in fact counts for 8 polyominos: each rotation and its mirror image.

Algorithm 1 `Algorithm X` (binary array A, subset of rows S)

```
if $A$ is empty then
  the problem is solved;
  print the solution set $S$;
else
  Choose a column, c (deterministically).
  Choose a row, $r$, such that $A[r, c] = 1$ (nondeterministically).
  Include $r$ in the partial solution $S$.
  for every $j$ such that $A[r, j] = 1$, do
    delete column $j$ from array $A$;
    for every $i$ such that $A[i, j] = 1$, do
      delete row $i$ from array $A$.
  Run Algorithm X (A, S).
```

Algorithm 2 `DLX` (binary array A, subset of rows S)

```
if $A$ is empty then
  the problem is solved;
  print the solution set $S$;
else
  Choose a column, c.
  CoverColumn (c).
  for every $r$ such that A[r,c]=1 do
    add $r$ to the partial solution set $S$.
    for every $j$ such that A[r,j]=1 do
      CoverColumn (j).
    DLX (A, S).
    for every $j$ such that A[r,j]=1 do
      UncoverColumn (j).
    remove $r$ from the partial solution set $S$
  UncoverColumn (c).
```

The terminology used above may be a bit obscure, and exactly how this can solve a polyomino tiling may not be exactly clear. We will restate the algorithm above in terms of polyomino tiling, which will make it a bit clearer. First, we will use the term *empty space* to refer the squares not yet covered by polyominos. At the start of the algorithm, the empty space E will be equal to the whole space that we would like to tile. We will use *placement* to refer to the combination of a polyomino and a position in the space to be tiled: a placement can either be *legal* (all of its squares are still in E, i.e. they haven't been covered) or *illegal* (some of its squares have already been covered). We will use L to refer to all remaining legal placements. A *partial tiling* will refer to a combination of legal placements within the space to be tiled: when the algorithm begins, the partial tiling P will be empty.

Algorithm 3 `PolyominoDLX` (empty space E, partial tiling P, legal placements L)

```
if $E$ is empty (there are no uncovered squares) then
    the partial tiling $P$ is now complete (it covers every original square);
    print $P$.
else
    Choose an uncovered square $s$ in $E$.
    for every legal placement $p$ in $L$ that covers $s$, do
        add $p$ to the partial tiling $P$;
        remove the squares covered by $p$ from the empty space $E$.
        for every placement $q$ in $L$ that was legal until $p$ was added do
            $q$ is now illegal, remove $q$ from $L$.
        PolyominoDLX (E, P, L).
        Remove $p$ from the partial tiling $P$;
        add the squares covered by $p$ back into $E$.
        for every placement $q$ that is now legal because $p$ has been removed do
            add $q$ back into $L$.
```

This is very much like what a person would do when trying to cover a space with rectangles by hand. The person would keep adding polyomino tiles until he either covers the entire space or makes a hole that no tile can fill. In the latter case, rather than starting over at the beginning, the person is likely to remove the last tile that he placed and try a different one in the same. If every tile that he tries in that spot results in a hole, then there must have been a problem with the tile that he placed before *that* one, and so on. The only difference is that `PolyominoDLX`, in the form stated above, will continue searching after it finds a valid tiling, in will continue until every valid tiling has been found. For a graphical representation of what `PoyominoDLX` is doing, see Figure 17, which demonstrates how the algorithm performs a depth-first search of the tree of all partial tilings, where one partial tiling is a descendant of another if it includes one additional tile.

The above algorithm can be easily modified to return only one tiling. We change `PolyominoDLX` so that it returns a boolean value. In the terminal case, where we are sure that we have found a tiling, it will return `true`. In other cases, the algorithm checks to see if the recursive call to `PolyominoDLX` has returned `true`, which means that a comlete tiling has been found that includes placement p. If this is the case, then the algorithm will immediately return `true` to the preceding instance of `PolyominoDLX`, which will immediately return `true`, etc. until the original call returns `true`. If none of the instances of `PolyominoDLX` return `true`, then no placement p that covers square s can result in a complete tiling. This must mean that there was an error in a preceding instance of `PolyominoDLX` (i.e. a bad choice of p at a previous step), so the algorithm returns false. If the original instance of `PolyominoDLX` returns `false`, then there is no way to tile the space with the polyominos that we have chosen to work with.

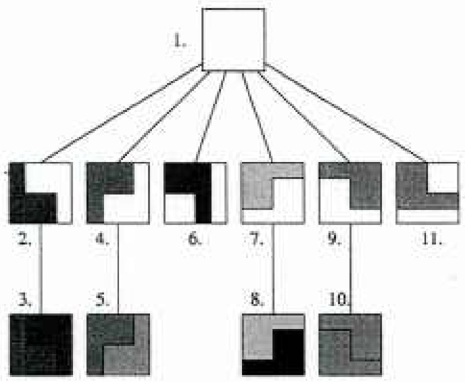


Figure 17: An example of a search tree created by `PolyominoDLX`. Each numbered domain represents a call to `PolyominoDLX`: the number indicates the order in which the call was created, and the tiles in the domain are the placements that were in P at the time `PolyominoDLX` was called.

6.2.1 Column Selection

Notice that in the description of `DLX`, no description is given of how the uncovered column c is chosen. In fact any method of choosing will work, but the speed of the algorithm depends on the choice made. After a choice of column c is made, we spawn a new instance of `PolyominoDLX` for every row such that $A[r, c] = 1$, and each instance represents a new branch in the search tree as imagined in Figure 17. If there are many branches to search at every step, a lot of time can be spent search branches that will not lead to a complete tiling. Therefore, a good idea is to go for a minimal number of branches at each step, i.e. to choose a column which has the fewest remaining entries. With this heuristic chosen, the behavior of `PolyominoDLX` becomes even more like the behavior of a human, because the squares with the fewest remaining legal placements will be the ones chosen, and those are the ones near the corners of the untiled space E. In other words, the algorithm will begin in a corner, and it will place new tiles next to tiles it has already placed.

6.2.2 Row Order

In addition, no description is given of the order in which the algorithm attempts to add rows r to the partial tiling P. One consequence of the data structure used in `DLX` is that instances of r such that $A[r, c] = 1$ can only be found in increasing order of r. A consequence of this for `PolyominoDLX` is that, at the beginning of the algorithm, each placement is given a unique number from 1 to the number of placements, and at every step, placements are attempted in increasing order. When we are trying to find all tilings, this is not a problem, but when we are trying to find only one, the order in which the placements are numbered affects the tiling that is found.

Consider the two small examples given in Figure 18. It's clear that each problem has the same two possible tiling. In problem 1, we will only find the first tiling, no matter the order in which we choose the squares. In problem 2, the order in which we choose the squares affects the tiling that we find: choosing square 1, we will find the first tiling; choosing square 2, we will find the second tiling. Thus we can say that the order in which we choose the squares has a limited effect on the tiling that we find, while the initial ordering of the placements has a much greater effect.

Algorithm 4 `boolean` = `PolyominoDLX` (empty space E, partial tiling P, legal placements L)

```
if $E$ is empty (there are no uncovered squares) then
  the partial tiling $P$ is now complete (it covers every original square);
  print $P$;
  return true.
else
  Choose an uncovered square $s$ in $E$.
  for every legal placement $p$ in $L$ that covers $s$, do
    add $p$ to the partial tiling $P$;
    remove the squares covered by $p$ from the empty space $E$.
    for every placement $q$ in $L$ that was legal until $p$ was added do
      $q$ is now illegal, remove $q$ from $L$.
    tf = PolyominoDLX (E, P, L).
    if tf is true then
      return true.
    Remove $p$ from the partial tiling $P$;
    add the squares covered by $p$ back into $E$.
    for every placement $q$ that is now legal because $p$ has been removed do
      add $q$ back into $L$.
  return false.
```

	▘	▝	▖	▗
▌	1	0	1	0
▐	0	1	0	1
▀	1	1	0	0
▄	0	0	1	1

	▘	▝	▖	▗
▌	1	0	1	0
▀	1	1	0	0
▄	0	0	1	1
▐	0	1	0	1

Figure 18: Problem 1 on the left and Problem 2 on the right have equivalent solutions, but different solutions will be found by `PolyominoDLX`.

6.3 Algorithms

6.3.1 Naïve Example

Suppose we want to satisfy the third criterion at the expense of the second. We can use `PolyominoDLX` in a straight forward fashion to guarantee that our algorithm produces all possible tilings with equal likelihood.

Algorithm 5 P = `NaïveAlg` (untiled space E)

```
1: Generate the set $L$ of all legal placements for the space $E$, and initialize the partial solution $P = \emptyset$.
2: Generate and store all possible tilings using PolyominoDLX($E, L, P$).
3: Select a tiling $P$ randomly from these tilings,
4: return $P$.
```

The problem with this idea is that the number of possible tilings, for every non-trivial set of polyominos, grows exponentially with the size of the domain. Consider the L-octomino used by Mailloux et al.: there are four ways that it can tile a 4×4 square. Therefore, if we wish to tile a $4n \times 4n$ square, we could simply divide it into n^2 smaller squares and tile each randomly, which means that there are at least $4^{n^2} = 16^n$ ways to tile the whole space. Given that the aperture they wish to tile is 64×64, there are at least 1.8×10^{19}

tilings which can be considered trivial tilings. The largest square that we have been able to generate all tilings for is 12×12.

6.4 Randomized PolyominoDLX

Given the importance of row order in DLX, the next algorithm that we propose is a randomized version of PolyominoDLX.

Algorithm 6 P = RandPolyDLX(untiled space E)

```
1: Generate the set $L$ of all legal placements for the space $E$, and initialize the partial solution $P = \emptyset$.
2: Shuffle the order of $L$.
3: Use PolyminoDLX($E, L, P$) to find only one tiling.
4: return $P$.
```

How does this algorithm compare to the naïve one in terms of how uniformly it generates tilings? First, it is clear that every complete tiling can be generate by this algorithm: all that is required is for the placements in the tiling to be shuffled to the beginning of the list, so that they will always be selected before other placements, regardless of the order in which we choose squares. Are all tilings generated with equal likelihood? The answer is no. Consider the two partial tilings in Figure 19: RandPolyDLX is as likely to begin with the two placements on the left as it is to begin with the two placements on the right. On the left, though, RandPolyDLX could go on to produce more than 64 tilings, while the pattern on the right results in only one complete tiling. Clearly, the algorithm is much more likely to produce the tiling on the right than one of the tilings on the left.

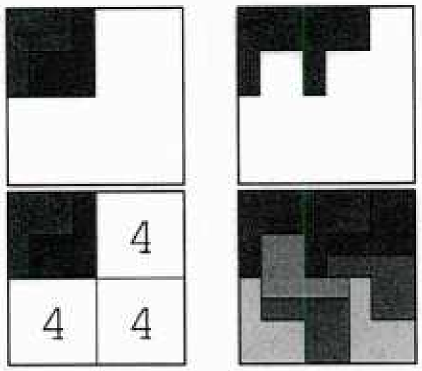


Figure 19: The two partial tilings on top are equally likely to occur during a call to RandPolyDLX, but the one on the left has only one descendant, while the one on the right has at least 64.

This algorithm is clearly more space efficient and time efficient than the naïve example above, but how fast is it? The time that it takes RandPolyDLX to find a complete tiling is directly proportional to the number of recursive instances of PolyominoDLX that are called before a complete tiling is found. Recall the search tree as presented in Figure 17: at each branching point of the tree, the randomized algorithm takes each descendant path with equal probability. If it attempts to go down a branch where there are no solutions, it must visit all of the nodes in that branch before it can try another branch.

Let us suppose that we are at node x, which has descendants $y_1, y_2, \cdots, y_m$, each of which has a solution somewhere in its branch, and descendants $z_1, z_2, \ldots, z_n$, each of which has no descendants in its branch. $p(y_i)$ will be the average number of nodes visited *after* y_i before finding a solution, and $t(z_j)$ will be the total number of nodes *below* y_j in the tree. So if y_i is a node that represents a complete tiling, then $p(y_i) = 0$; if

y_i has one descendant, which happens to represent a final solution, then $p(y_i) = 1$, etc. Let $\bar{p}$ be the average of the $p(y_i)$ values and $\bar{t}$ be the average of the $t(z_j)$ values. The value of $p(x)$ will be

$$\begin{aligned} p(x) &= \sum_{j=0}^{n} (\text{probability that } j \text{ of the } z \text{ nodes are visited before a } y \text{ node}) * (j(\bar{t}+1) + (\bar{p}+1)) \\ &= \sum_{j=0}^{n} \left(\frac{(m+n-(j+1))!}{(m+n)!} m \frac{n!}{(n-j)!} \right) * (j(\bar{t}+1) + (\bar{p}+1)). \end{aligned}$$

Because of the recursive nature of this relationship, we can compute the average number of steps used by RandPolyDLX at the same time that we ask PolyominoDLX to find all tilings. We have computed this value for tilings of size 4×4, 8×8, and 12×12 using the L-octomino: computing it for 16×16 would be a batch job that would take days. The average number of instances that we computed for each were 2.4, ~ 9.7, and ~ 29.3, respectively.

If we assume that the growth in the average number of instances is $\mathcal{O}(n^p)$ for a square of size $n \times n$, we know that p must be at least 2, because at least one instance must be created for each tile in the complete tiling, and the number of tiles in a complete tiling grows quadratically. The growth from 2.4 to 9.7 suggests an exponent of $p \approx 2.02$, while the growth from 9.7 to 29.3 suggests an exponent of $p \approx 2.715$. If we assume that the growth is $\mathcal{O}(e^{n^p})$, then the change from 2.4 to 9.7 suggests $p \approx 1.38$, while the change from 9.7 to 29.3 suggests $p \approx 0.97$. We only have three data points to work with, but they seem to suggest that the average run-time of the algorithm is super-polynomial, but sub-exponential. To understand the behavior of the run-time at larger n, we have to turn to empirical data.

Experiments show that although the median compute time is small in most instances, there are cases where the program takes unacceptably long time to terminate. No matter our estimation of the growth of the average run-time, the existence of pathological instances where the algorithm essentially runs forever means that the algorithm in its current form does not scale well enough for it to be used for larger n: the largest square for which the algorithm is reliable is 16×16. One modification, though, will extend reliability to another order of magnitude.

6.4.1 Restarting

Given the large gap between the median run-time and the pathological cases for $n = 32$, RandPolyDLX is likely to find a tiling faster if it restarts after its run-time passes a certain cut-off point. To achieve this we make a few small modifications to PolyominoDLX: we include in its arguments the start time t of the program, which we pass unchanged to every recursive call, and a maximum runtime Δ. Then, at the start of a call to PolyominoDLX, we find the current time T and check to see if $T - t < \Delta$. If it isn't, then we return a restart flag to the preceding instance of the algorithm, which returns the restart flag to its predecessor, and so on. The new algorithm looks like this:

Algorithm 7 $P =$ RestartDLX (untiled space E, maxtime Δ)

```
1: Generate the set $L$ of all legal placements for the space $E$.
2: while PolyominoDLX returns a restart flag do
3:     Set $P = \emptyset$.
4:     Shuffle the order of $L$.
5:     Record the current time $t$.
6:     PolyminoDLX($E, L, P, t, \Delta$).
7: return $P$.
```

Suppose there is a set probability q such that we want to minimize the time T such that with probability q, RestartDLX runs in less time than T, and suppose that the CDF $f(t)$ for the average run-time of RandPolyDLX is known. Then for a given Δ, the CDF $g(t)$ for RestartDLX(Δ) is

$$g(t) = 1 - (1 - f(\Delta))^{\lfloor \frac{t}{\Delta} \rfloor} + f(t - \Delta\lfloor\frac{t}{\Delta}\rfloor)(1 - f(\Delta))^{\lfloor \frac{t}{\Delta} \rfloor}.$$

We can ignore the second part by assuming that t is some multiple k of Δ, in which case

$$g(k\Delta) = 1 - (1 - f(\Delta))^k.$$

We want to find the smallest value of $k\Delta$ such that $g(k\Delta) > q$. This is equivalent to finding the smallest value of $k\Delta$ such that $(1 - f(\Delta))^k < 1 - q$.

Restarting RandPolyDLX will not eliminate any tiling from being generated: as stated above, the right ordering of the placements will result in a given tiling being found without any backtracking. It may be true, though, that restarting changes the probability that a given tiling will be generated. For every tiling P, we could theoretically enumerate all of the of ways that RandPolyDLX could find P, and find the distribution of how long it would take to find P in each case. If a relatively high proportion of those times are longer than the cut-off Δ, then RestartDLX would be less likely to find P than RandPolyDLX. In my work to develop a metric for the disorder in a tiling, the outputs of RestartDLX seem to be equally disordered for different values of Δ, which suggests that the effect of restarting is fairly small.

6.4.2 Rough edges

It is intuitive to think that a lot of the time in the above algorithms is spent making the tilings conform to the edges of the area being tiled; indeed, Mailloux et al. state that the were never able to tile a large domain exactly by hand, and instead allowed the polyominos to hang over the edge, so long as the entirety of the aperture was covered. We can make a few slight changes to PolyominoDLX that will allow it to operate in the same fashion. First, we give every square a flag, marking it as either "necessary" or "unnecessary". Next, we change the stopping criterion of PolyominoDLX: whereas before it stopped when there were no uncovered squares, it now stops when there are no uncovered necessary squares. Finally, when selecting a square s in the main stage, only necessary squares are considered. The rough algorithm looks like this:

Algorithm 8 P = RoughDLX(untiled space E)

```
1: Mark every square in $E$ as necessary.
2: Create a buffer $B$ of unnecessary squares around $E$ such that every polyomino placement that contains
   one square of $E$ is contained in $B \cup E$.
3: $P$ = RandPolyDLX($B \cup E$).
4: **return** $P$.
```

Computational experiments show that the algorithm has fewer pathological cases at $n = 64$ than RandPolyDLX does, but the pathological cases still exist. It seems that allowing rough edges is not a fix that makes all n more tractable: for $n = 128$, the pathological cases are still too frequent to make the algorithm practical.

6.5 Pair-Swapping

There is something counterintuitive about using variants of DLX to create the sort of polyomino tilings that uses a versatile polyomino set, such as the set of all tetrominos, over a large area. DLX was created to answer the decision question, "can this space be tiled"? When tiling a 64×64 square—or any rectangle where both lengths are even—with tetrominos, the decision question is not a difficult one: the space can be filled with 2×2 squares. The desire for an algorithm that takes advantage of an easily identified tiling resulted in the PairSwapping algorithm.

Algorithm 9 $P =$ `PairSwapping`)untiled space E

```
Initialize $P$ to be the easily identified tiling of $E$.
for as long as is desired do
  choose a placement $p$ at random
  if $p$ has a neighbor placement $q$ such that the profile of $p$ and $q$ can be filled with two different placements, $\hat{p}$ and $\hat{q}$ then
    remove $p$ and $q$ from $P$ and add $\hat{p}$ and $\hat{q}$.
return $P$.
```

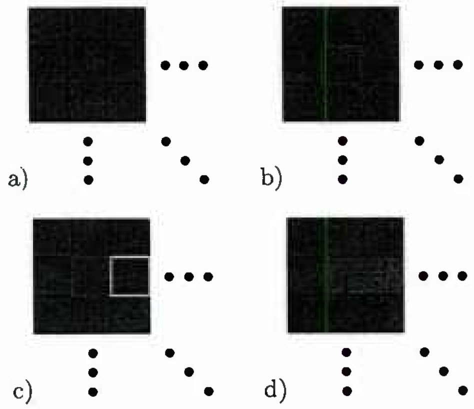


Figure 20: A graphical demonstration of the pair-swapping process.

The advantage of this algorithm is that swapping two tiles is a constant-time procedure, and that P is a complete tiling throughout the algorithm. Instead of a search tree for finding a complete tiling, we now picture a web of complete tilings, with two tilings being adjacent to each other if one can be transformed into the other through one pair-swap. Using this metaphor, `PairSwapping` is just a random walk within this web.

We know *a priori* that the run-time for this algorithm scales well: in fact, we know that it scales as $\mathcal{O}(n^2)$ for an $n \times n$ square. Whatever criteria we use to determine how many swaps k necessary (on average) for an $n \times n$ square, the same criteria will find that $4k$ swaps are necessary (on average) for a $2n \times 2n$ square, since it has four times the area.

An interesting question is whether every possible tiling can be reached from the initial tiling using pair swaps. This is clearly dependent on the set of polyominos being used, and in most cases is a difficult question to tackle. Suppose we limit ourselves to a complete set of polyominos, i.e. a set of all polyominos of a given size k. Though we omit the proof here, it can be shown that any tiling of a rectangle with dominos can be transformed into any other tiling. Even in the case of tetrominos, we have yet to come up with an answer to this question.

Regardless of the answer to the above question, we can still know something about the likelihood of being reached for any tiling in the orbit of the initial tiling. It helps to think of a chemical reaction, with configurations of tiles being certain states that can change into each other with equal forward and backward probability. Those configurations with more reactions, i.e. more ways that other configurations can be transformed into each other, will be more prevalent even as the reactions are allowed to continue on infinitely. So in the case of tetrominos, for example, if we begin with a field of 2×2 blocks, and allow pair-swapping to go on forever, we will almost always see a high prevalence of interlocking L's than we would see in a tiling created by `DLX`, as demonstrated in Figure 21.

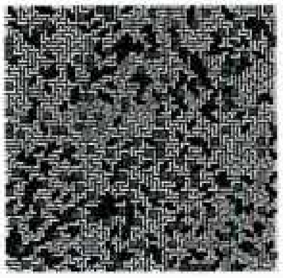


Figure 21: An example of a tiling produced by PairSwapping.

The flaw of this method is that it does not meet the first criterion from the introduction. It works well for the set of all tetrominos because a single profile can be covered by many pairs of tiles. In the case of the L-octomino, however, the only profile that allows more than one pair of tiles is a 4×4 square, which means that any initial tiling can only be trivially changed.

6.6 Hybrid Swap

The algorithm HybridSwap generalizes the idea of pair-swapping to any set of polyominos, while keeping its good run-time properties. For the set of polyominos in question, we choose a stencil size k and a cut-off time Δ, such that RestartDLX can tile a $k \times k$ square with the polyominos in a reasonable time with high probability. The algorithm then looks like this:

Algorithm 10 P = HybridSwap (untiled area E, cut-off time Δ, stencil size k)

```
Initialize $P$ to be the easily identified tiling of $E$.
for as long as is desired do
  Choose a $k \times k$ square inside $E$ according to some method.
  Set the untiled space $F = \emptyset$.
  for every tile $p$ in $P$ such that $p$ lies entirely with the chosen square do
    remove $p$ from $P$.
    add the squares covered by $p$ to $F$.
  $Q$ = RestartDLX(F,$\Delta$).
  $P = P \cup Q$.
return $P$.
```

Because k is independent of the size of the original space E, and because the run-time of RestartDLX is bounded above with high probability, each iteration of the main for-loop takes essentially constant time. Therefore, regardless of the criteria that determine where to place the stencil, the run-time of HybridSwap is $\mathcal{O}(n^2)$. A good criteria is to place the stencil at $k/2$ intervals throughout the space, so that whenever there is an edge between altered and unaltered tiles, that edge is then taken out by a subsequent use of the stencil. The stencil can also be placed randomly throughout the space, until one is satisfied that every square has been retiled at least once. Initial results using the disorder metric described below indicate that these two methods produce tilings that qualitatively that same.

The probability that a given tiling will be generated by HybridSwap is a murky subject. Given an $n \times n$ square and a set of polyominos, the question one would like to have answered is: for which stencil sizes k can HybridSwap transform the initial tiling into every possible tiling? Its clear that it can for $k = n$, because then HybridSwap is essentially RestartDLX. For the L-octomino, we have found patterns such that an 8×8 stencil cannot change in any non-trivial way, which means they cannot be reached from the initial tiling, but we have no results for larger k. The next question is whether the tilings produced by HybridSwap are qualitatively different from those produced by RestartDLX. In some cases, such as using a 32×32 stencil on a 64×64 space, the disorder metric was able to differentiate between the averaged output of HybridSwap and RestartDLX, but the difference appeared to be slight.

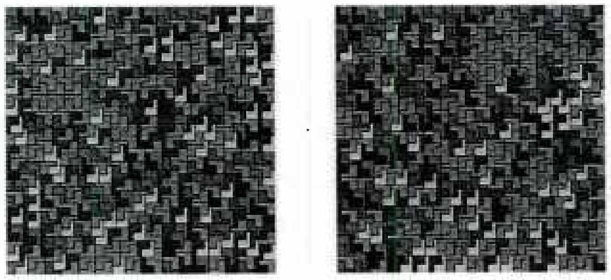


Figure 22: The tiling on the left was created with RestartDLX; the tiling on the right was created by HybridSwap.

6.7 Disorder Metric

The quantization lobes described by Mailloux at al. are the result of periodicity: when there is some naturally occurring period in the layout of the subarrays, aliasing between that period and the wavelength degrades the signal. The idea behind a disorder metric is that a maximally disordered tiling should exhibit no significant periodicity in any direction. The human eye is remarkably adept at picking out order, so we propose the following transformation of a polyomino tiling that allows one to better spot any periodic behavior:

```
Algorithm 11 (array $I$) = DisorderImage($m \times n$ space $E$, tiling $P$)
Set $I$ to be an array of zeros of size $2m - 1 \times 2n - 1$
for $i \in \{-(m-1), \cdots, m-1\}$ do
  for $j \in \{-(n-1), cdots, n-1\}$ do
    for every tile $p$ in $P$ do
      if There is a tile identical to $p$, but shifted down by $i$ and right by $j$ then
        $I[m+i, n+j] += 1$.
```

See Figure 23 for a graphical depiction of the process being described.

The array I will have large values where a large number of tiles can be shifted by the same amount to exactly reproduce a different section of the space. When the array I is viewed using either color or a third dimension to differentiate high and low values, it becomes easier to spot where periodicity occurs. Perfect periodicity would look like a line radiating from the center of the image, along which there are many values which are higher than their surroundings.

One strong point of this method is that it is general enough to be applied to arbitrary size domains, and to arbitrary sets of polyominos. For a given set of polyominos, though, the tiles may tend to certain patterns that allow one to distill the image I into a single value. For the L-octomino, a domain composed entirely of squares, as in Figure 24 (a), produces an image as in Figure 24 (b), where all of the non zero values are found at indices $[i, j]$ such that $m - i \equiv n - j \equiv 0 \mod 4$. In general, every tiling will have higher values at these indices than in the surrounding area, but the difference is noticeably slighter in disordered tilings (see Figure 24 (c) and (d)). We can even boil the array I down to a single value by summing its values at 0

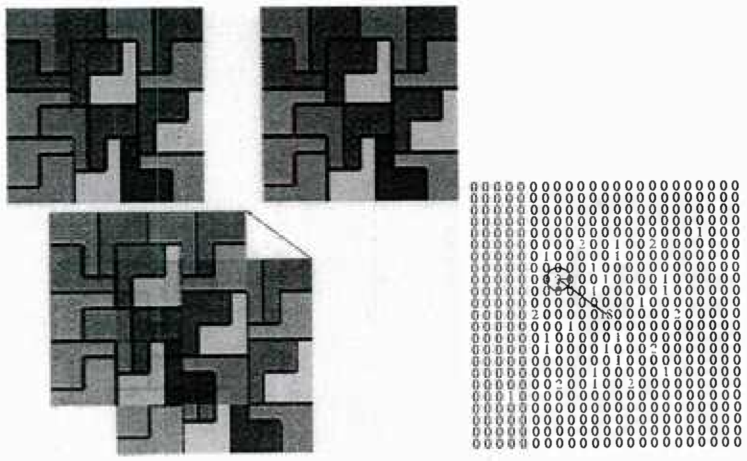


Figure 23: This a 12 × 12 array. When we lay one copy on top of the other, and then shift if to the left by 4 squares and up by 3, we see that one red tiles exactly overlays another red tile and one yellow tile exactly overlays another yellow tile. Because (12, 12) is taken to be the center of the image I that will be create, we can say that $I(12-4, 12-3) = 2$. Notice that $I(12, 12)$, which represents zero-offset, is equal to 18: this is because at zero-offset, every tile overlays itself.

mod 4 indices:

$$x = \sum_{i \in \mathcal{I}(m)} \sum_{j \in \mathcal{I}(n)} I[m+i, n+j],$$

where $\mathcal{I}(m) = \{-(m-4), -(m-8), \cdots, -4, 0, 4, \cdots, (m-8), (m-4)\}$.

Given two arrays, the one with the lower x value should be the more disordered, and thus the less periodic of the two. This value was able to correctly rank the best and worst performing arrays (in terms of sidelobe power) generated by Mailloux et al. using their previous software (see Figure 25). Work is currently underway to confirm this correlation on other arrays.

7 Participants

The project participants were

- PI: Tim Warburton, Rice University
- Post-doc: Tanya Vdovina
- Student: Toby Isaac (Graduated 2008, US citizen). Currently a graduate student at ICES, UT Austin
- Consultant: Professor Thomas Hagstrom, Southern Methodist University
- Collaborator: Jan S. Hesthaven, Brown University
- Collaborator: Andreas Klöckner (Graduate Student, Brown University)
- Collaborator: Nico Gödel (Graduate Student, University of Hamburg)
- Collaborator: Steffen Schomann (Graduate Student, University of Hamburg)
- AFRL Contacts: Thomas Roberts, Robert Mailloux, and Scott Santarell, Hascom AFB

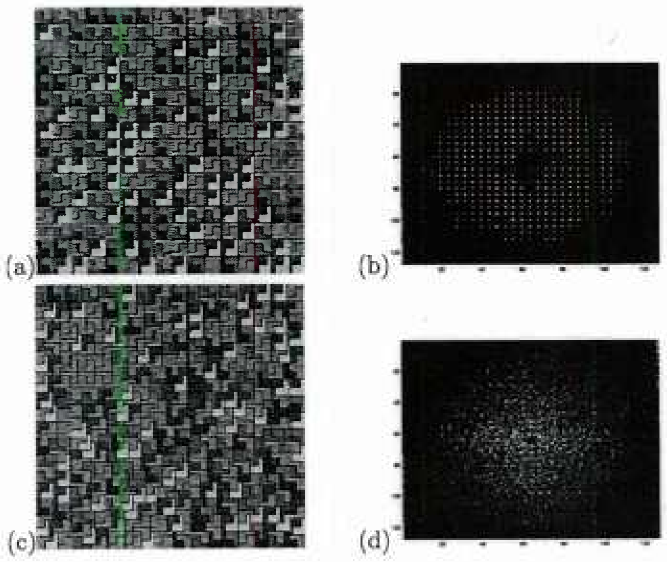


Figure 24: A comparison of the images generates by a domain divided into squares and a typical output of `RestartDLX`.

8 Presentations

Conference Talks:

- Oberwolfach Conference on Non-traditional Finite Element Methods, Mathematisches Forschungsinstitut Oberwolfach, Germany, 2008,
- Discontinuous Galerkin Methods for Partial Differential Equations, Banff International Research Station, Canada, 2007.
- The 8th International Conference on Mathematical and Numerical Aspects of Waves, Reading, UK, 2007.
- 6th International Congress on Industrial and Applied Mathematics, Zurich, Switzerland, 2007.
- International Workshop on High-Order Finite Element Methods, Herrsching, Germany, 2007.
- Oberwolfach Conference on Computational Electromagnetism and Acoustics, Mathematisches Forschungsinstitut Oberwolfach, Germany, 2007.
- 7'th World Congress on Computational Mechanics, Los Angeles, CA, USA, 2006
- SIAM Annual Meeting, Boston MA, 2006
- Advances in Computational Scattering BIRS, Banff, Canada, 2006.

Departmental Seminars:

- Center for Computation and Technology Colloquium, Louisiana State University, LA, 2008.
- Mathematics and Computer Science Division, Argonne National Laboratory, Chicago, USA, 2007.

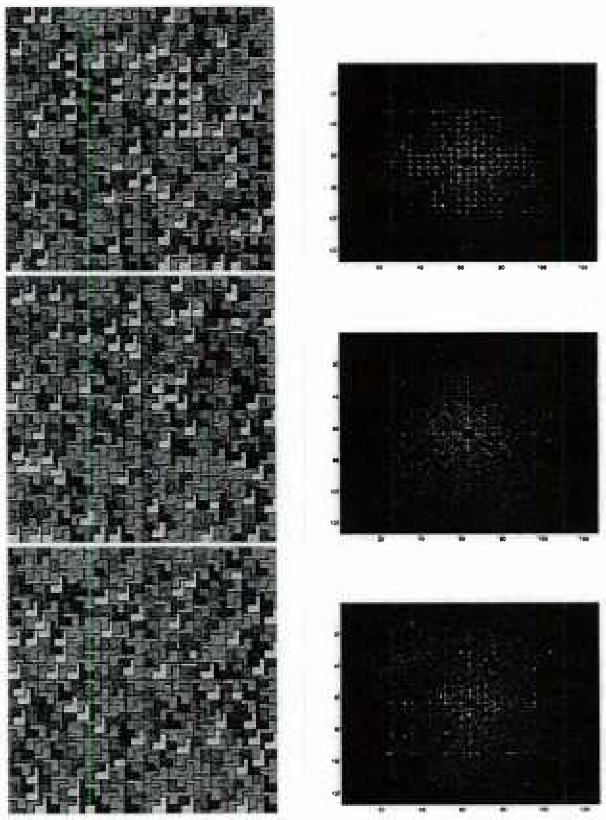


Figure 25: The arrays presented by Mailloux et al. and the images that they generate. Notice the higher values at mod 4 indices for the first image. (Colormaps are uniform across the images.)

- Computer and Information Technology Institute, Rice University, Houston, TX, 2007.
- Department of Mathematics, Rice University, Houston, TX, 2006.
- ICES, U. Texas at Austin, TX, USA 2006.
- Scientific Computing Seminar, University of Houston, Houston, TX, USA, 2006.

9 Publications

The following list includes published journal articles.

1. *Convergence analysis of an adaptive interior penalty discontinuous Galerkin method*, R. H. W. Hoppe, G. Kanschat, and T. Warburton. SIAM Journal on Numerical Analysis 2008. to appear.
2. *Radiation boundary conditions for time-dependent waves based on complete plane wave expansions*, Thomas Hagstrom, Timothy Warburton, and Dan Givoli, Journal of Computational and Applied Mathematics, 2008, to appear.

3. *Taming the CFL Number for Discontinuous Galerkin Methods on Structured Meshes*, T. Warburton and T. Hagstrom, SIAM Journal on Numerical Analysis, Volume 46, Issue 6, pp. 3151-3180 2008.

4. *An Explicit Construction for Interpolation Nodes on the Simplex*, T. Warburton, Journal of Engineering Mathematics, Volume 56, Number 3, pp. 247-262, November, 2006.

The following lists conference abstracts and student thesis work derived directly and indirectly from this project.

1. *Discontinuous Galerkin Methods for High Frequency Electromagnetic Computations*, N. Gdel, S. Schomann, T. Warburton, M. Clemens Eucap 2009, 3rd European Conference on Antennas and Propagation - 23-27 March 2009 in Berlin, Germany. Abstract accepted for presentation.

2. *Numerische Simulation hochfrequenter elektromagnetischer Felder mit der Discontinuous Galerkin Finite Elemente Methode [Numerical Simulations of High-Frequency Electromagnetic Fields with the Discontinuous Galerkin Finite Element Method]*, N. Gdel, M. Clemens, U.R.S.I. 2008 Kleinheubacher Tagung (KH 2008), Miltenberg, 22.-25.09.2008. Abstract accepted for presentation.

3. *Lokale Zeitintegrationsverfahren zur effizienten Berechnung hochfrequenter elektromagnetischer Felder mit der Discontinuous Galerkin Finite Elemente Methode [Local Time Integration for Efficient Computation of High-Frequency Electromagnetic Fields with the Discontinuous Galerkin Finite Element Method]* , S. Schomann, N. Gdel, M. Clemens, U.R.S.I. 2008 Kleinheubacher Tagung (KH 2008), Miltenberg, 22.-25.09.2008. Abstract accepted for presentation.

4. *Lokale Zeitintegrationsverfahren zur effizienten Berechnung hochfrequenter elektromagnetischer Felder mit der Discontinuous Galerkin Finite Elemente Methode [Local Time Integration for Efficient Computation of High-Frequency Electromagnetic Fields with the Discontinuous Galerkin Finite Element Method]* , S. Schomann Student research project report, Helmut-Schmidt-University Hamburg, Germany, 2008.

5. *Accelerating the Discontinuous Galerkin Time-Domain Method*, Timothy Warburton, MFO Report Nonstandard Finite Element Methods, No. 36/2008.

6. *On Complete Radiation Boundary Conditions and Optimal Absorbing Layers*, T. Hagstrom, and T. Warburton, , 8th International Conference on Mathematical and Numerical Aspects of Waves.

7. *Taming the CFL Condition for Discontinuous Galerkin in Two-Dimensions*, T. Warburton, and Thomas Hagstrom, 8th International Conference on Mathematical and Numerical Aspects of Waves.

8. *A Survey of Discontinuous Galerkion Methods for Time-Domain Electromagnetics*, T. Warburton, Oberwolfach Conference on Computational Electromagnetism and Acoustics, 2007.

10 Transitions

As noted in the text we have worked with Adour Kabakian, of Hypercomp Inc, to transfer the multirate time stepping algorithm. He has implemented it within one of their DGTD solvers and sees noticeable speeds ups for sample problems of interest.

We also worked with Mailloux's group at Hanscomb AFB. Toby Isaac visited them and shared his implementation of the tiling algorithm discussed in this report. They reported that antennae tilings derived from this algorithm delivered improved side-lobe reduction. Based on feedback from this group we added the facility to use user defined tiles in creating these tilings.

References

[1] M. Ainsworth. Dispersive and dissipative behaviour of high order discontinuous Galerkin finite element methods. *J. Comp. Phys.*, 198:106–130, 2004.

[2] R. Biswas, K. Devine, and J. Flaherty. Parallel, adaptive finite element methods for conservation laws. *Appl. Num. Math.*, 14(1–3):255–283, 1994.

[3] P. Borwein and T. Erdelyi. *Polynomials and Polynomial Inequalities.* Graduate Texts in Mathematics. Springer-Verlag, New York, first edition, 1995.

[4] C. Canuto, M. Hussaini, A. Quarteroni, and T. Zang. *Spectral Methods in Fluid Dynamics.* Springer-Verlag, New York, 2 edition, 1990.

[5] E. Chung and B. Engquist. Optimal discontinuous galerkin methods for wave propagation. *SIAM J. Num. Anal.*, 44:2131–2158, 2006.

[6] B. Cockburn, S. Hou, and C.-W. Shu. The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: the multidimensional case. *Math. Comp.*, 54:545–581, 1990.

[7] B. Cockburn, S.-Y. Lin, and C.-W. Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: one dimensional systems. *J. Comp. Phys.*, 84:90–113, 1989.

[8] B. Cockburn and C.-W. Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws II: general framework. *Math. Comp.*, 52:411–435, 1989.

[9] B. Cockburn and C.-W. Shu. The local discontinuous Galerkin method for time-dependent convection-diffusion systems. *SIAM J. Num. Anal.*, 35(6):2440–2463, 1998.

[10] B. Cockburn and C.-W. Shu. The Runge-Kutta discontinuous Galerkin method for conservation laws V: multidimensional systems. *J. Comp. Phys*, 141:199–224, 1998.

[11] B. Cockburn and C.-W. Shu. Runge-Kutta discontinuous Galerkin methods for convection-dominated problems. *J. Sci. Comp.*, 16:173–261, 2001.

[12] J. Goodrich, T. Hagstrom, and J. Lorenz. Hermite methods for hyperbolic initial-boundary value problems. *Math. Comp.*, 75:595–630, 2006.

[13] D. Gottlieb and E. Tadmor. The CFL condition for spectral approximations to hyperbolic initial-boundary value problems. *Math. Comp.*, 56:565–588, 1991.

[14] J. S. Hesthaven and T. Warburton. High-order nodal methods on unstructured grids. i: Time-domain solution of Maxwell equations. *J. Comp. Phys.*, 181:186–221, 2002.

[15] J. S. Hesthaven and T. Warburton. High-order accurate methods for time-domain electomagnetics. *Comput. Model. Eng. Sci.*, 5(5):395–408, 2004.

[16] J. S. Hesthaven and T. Warburton. High-order nodal discontinuous Galerkin methods for the Maxwell eigenvalue problem. *Phil. Trans. Roy. Soc. London A*, 362:493–524, 2004.

[17] F. Hu and H. Atkins. Eigensolution analysis of the discontinuous Galerkin method with non-uniform grids. Part I: One space dimension. *J. Comp. Phys.*, 182:516–545, 2002.

[18] C. Johnson and J. Pitkäranta. Convergence of a fully discrete scheme for two-dimensional neutron transport. *SIAM J. Num. Anal.*, 20:951 – 966, 1983.

[19] D. A. Kopriva, S. L. Woodruff, and M. Y. Hussaini. Discontinuous spectral element approximation of Maxwell's equations. In B. Cockburn, G. E. Karniadakis, and C. W. Shu, editors, *Discontinuous Galerkin Methods: Theory, Computation and Applications*, volume 11 of *Lecture Notes in Computational Science and Engineering*, pages 355–361, New York, 2000. Springer-Verlag.

[20] D. A. Kopriva, S. L. Woodruff, and M. Y. Hussaini. Computation of electromagnetic scattering with a non-conforming discontinuous spectral element method. *Int. J. Num. Meth. Eng.*, 53:105–122, 2002.

[21] P. Lesaint and P. Raviart. On a finite element method for solving the neutron transport equation. In *Mathematical aspects of finite elements in partial differential equations*, New York, 1974. Math. Res. Center, Univ. of Wisconsin-Madison.

[22] T. Peterson. A note on the convergence of the discontinuous Galerkin method for a scalar hyperbolic equation. *SIAM J. Numer. Anal.*, 28(1):133–140, 1991.

[23] W. Reed and T. Hill. Triangular mesh methods for the neutron transport equation. Technical report, Los Alamos, Los Alamos, NM, 1973.

[24] G. Richter. An optimal-order error estimate for the discontinuous Galerkin method. *Math. Comp.*, 50:75–88, 1988.

[25] T. Warburton. Application of the discontinuous Galerkin method to Maxwell's equations using unstructured polymorphic hp-finite elements. In B. Cockburn, G. E. Karniadakis, and C. W. Shu, editors, *Discontinuous Galerkin Methods: Theory, Computation and Applications*, volume 11 of *Lecture Notes in Computational Science and Engineering*, pages 451–458, New York, 2000. Springer-Verlag.

[26] T. Warburton and M. Embree. The role of the penalty in the local discontinuous Galerkin method for Maxwell's eigenvalue problem. *Comp. Meth. Appl. Mech. Eng.*, 195:3205–3223, 2005.

[27] Joseph E. Flaherty, Raymond M. Loy, Mark S. Shephard, Boleslaw K. Szymanski, James D. Teresco, and Louis H. Ziantz. *Adaptive local refinement with octree load balancing for the parallel solution of three-dimensional conservation laws*, Journal of Parallel and Distributed Computing, 47(2):139–152, 1997.

[28] Jean-François Remacle, Katia Pinchedez, Joseph E. Flaherty, and Mark S. Shephard. *An efficient local time stepping-discontinuous galerkin scheme for adaptive transient computations*, Technical Report 2001-13, SCOREC, RPI, Troy, NY, 2001.

[29] Clint Dawson and Robert Kirby. *High resolution schemes for conservation laws with locally varying time steps*, SIAM Journal on Scientific Computing, 22(6):2256–2281, 2000.

[30] Serge Piperno. *Symplectic local time-stepping in non-dissipative DGTD methods applied to wave propagation problems*, ESIAM: Mathematical Modelling and Numerical Analysis, 40(5):815–841, 2006.

[31] Michael Dumbser, Martin Käser, and Eleuterio F. Toro. *An arbitrary high-order Discontinuous Galerkin method for elastic waves on unstructured meshes - V. Local time stepping and p-adaptivity*, Geophysical Journal International, 171(2):695–717, November 2007.

[32] Juien Diaz, and Marcus J. Grote. *Energy Conserving Explicit Local Time-Stepping for Second-Order Wave Equations*, SIAM Journal Scientific on Computing, in press 2007.

[33] C.W. Gear, D.R. Wells. *Multirate linear multistep methods* BIT Numerical Methods, 24:484-502, 1984.

[34] Jan.S. Hesthaven and Tim Warburton *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*, Springer Texts in Applied Mathematics, Vol. 54, 2008.

[35] S. Abarbanel and D. Gottlieb. On the construction and analysis of absorbing layers in CEM. *Applied Numerical Mathematics*, 27:331–340, 1998.

[36] I. Buck, T. Foley, D. Horn, J. Sugerman, K. Fatahalian, M. Houston, and P. Hanrahan. Brook for GPUs: stream computing on graphics hardware. In *International Conference on Computer Graphics and Interactive Techniques*, pages 777–786. ACM New York, NY, USA, 2004.

[37] M. H. Carpenter and C. A. Kennedy. Fourth-order 2N-storage Runge-Kutta schemes. Technical report, NASA Langley Research Center, 1994.

[38] B. Cockburn, S. Hou, and C. W. Shu. The Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Method for Conservation Laws. IV: The Multidimensional Case. *Mathematics of Computation*, 54:545–581, 1990.

[39] International Electrotechnical Commission. Letter symbols to be used in electrical technology - Part 2: Telecommunications and electronics. Technical report, International Electrotechnical Commission, Geneva, Switzerland, November 2000.

[40] D. Göddeke, R. Strzodka, and S. Turek. Accelerating double precision FEM simulations with GPUs. In *Proceedings of ASIM*, 2005.

[41] Nail A. Gumerov and Ramani Duraiswami. Fast multipole methods on graphics processors. *Journal of Computational Physics*, 227:8290–8313, September 2008.

[42] J. S. Hesthaven, S. Gottlieb, and D. Gottlieb. *Spectral Methods for Time-Dependent Problems.* Cambridge UniversityPress, 2007.

[43] J. D. Jackson. *Classical Electrodynamics.* Wiley, third edition, July 1998.

[44] G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing*, 20:359–392, 1999.

[45] S.E. Krakiwsky, L.E. Turner, and M.M. Okoniewski. Acceleration of finite-difference time-domain (FDTD) using graphics processor units (GPU). In *Microwave Symposium Digest, 2004 IEEE MTT-S International*, volume 2, pages 1033–1036 Vol.2, 2004.

[46] W. Li, X. Wei, and A. Kaufman. Implementing Lattice Boltzmann computation on graphics hardware. *The Visual Computer*, 19:444–456, 2003.

[47] E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym. Nvidia Tesla: A Unified Graphics and Computing Architecture. *Micro, IEEE*, 28:39–55, 2008.

[48] Nvidia Corporation. *NVIDIA CUDA 2.0 Compute Unified Device Architecture Programming Guide.* Nvidia Corporation, Santa Clara, USA, June 2008.

[49] W. H. Reed and T. R. Hill. Triangular mesh methods for the neutron transport equation. Technical report, Los Alamos Scientific Laboratory, Los Alamos, 1973.

[50] H. Si and K. Gaertner. Meshing Piecewise Linear Complexes by Constrained Delaunay Tetrahedralizations. In *Proceedings of the 14th International Meshing Roundtable*, pages 147–163. Springer, 2005.

[51] J. Stratton, S. Stone, and W. Hwu. MCUDA: An Efficient Implementation of CUDA Kernels on Multicores. Technical report, University of Illinois at Urbana-Champaign, Urbana-Champaign, IL, USA, March 2008.

[52] T. Warburton. An explicit construction of interpolation nodes on the simplex. *Journal of Engineering Mathematics*, 56:247–262, 2006.

[53] T. Warburton and T. Hagstrom. Taming the CFL Number for Discontinuous Galerkin Methods on Structured Meshes. *SIAM Journal on Numerical Analysis*, 46:3151–3180, 2008.

[54] R. C. Whaley, A. Petitet, and J. J. Dongarra. Automated empirical optimizations of software and the ATLAS project. *Parallel Computing*, 27:3–35, 2001.

[55] Wikipedia. Comparison of Nvidia graphics processing units — Wikipedia, The Free Encyclopedia, 2008. [Online; accessed 9-November-2008].

[56] B. Alpert, L. Greengard, and T. Hagstrom, *Rapid evaluation of nonreflecting boundary kernels for time-domain wave propagation*, SIAM J. Numer. Anal., 37 (2000), pp. 1138–1164.

[57] ——, *Nonreflecting boundary conditions for the time-dependent wave equation*, J. Comput. Phys., 180 (2002), pp. 270–296.

[58] D. Appelö, T. Hagstrom, and G. Kreiss, *Perfectly matched layers for hyperbolic systems: General formulation, well-posedness and stability*, SIAM J. Appl. Math., 67 (2006), pp. 1–23.

[59] S. Asvadurov, V. Druskin, M. Guddati, and L. Knizherman, *On optimal finite difference approximation of PML*, SIAM J. Numer. Anal., 41 (2003), pp. 287–305.

[60] E. Bécache, S. Fauqueux, and P. Joly, *Stability of perfectly matched layers, group velocities, and anisotropic waves*, J. Comput. Phys., 188 (2003), pp. 399–433.

[61] J.-P. Bérenger, *A perfectly matched layer for the absorption of electromagnetic waves*, J. Comput. Phys., 114 (1994), pp. 185–200.

[62] F. Collino, *High order absorbing boundary conditions for wave propagation models. Straight line boundary and corner cases*, in Proceedings of 2nd Int. Conf. on Math. and Numer. Aspects of Wave Prop. Phen., R. K. et al., ed., SIAM, 1993, pp. 161–171.

[63] A. de Hoop, P. van den Berg, and R. Remis, *Absorbing boundary conditions and perfectly matched layers - an analytic time-domain performance analysis*, IEEE Trans. on Magnetics, 38 (2002), pp. 657–660.

[64] J. Diaz and P. Joly, *An analysis of higher-order boundary conditions for the wave equation*, SIAM J. Appl. Math., 65 (2005), pp. 1547–1575.

[65] ——, *A time-domain analysis of PML models in acoustics*, Computer Meth. Appl. Mech. Engrg., 195 (2006), pp. 3820–3853.

[66] V. Druskin, M. Guddati, and T. Hagstrom, *On generalized PML optimized for propagative and evanescent waves.* In preparation, 2008.

[67] A. Ergin, B. Shanker, and E. Michielssen, *Fast evaluation of three-dimensional transient wave fields using diagonal translation operators*, J. Comput. Phys., 146 (1998), pp. 157–180.

[68] D. Givoli, T. Hagstrom, and I. Patlashenko, *Finite element formulation with high order absorbing boundary conditions for time-dependent waves*, Comput. Meth. Appl. Mech., 195 (2006), pp. 3666–3690.

[69] M. Guddati and K.-W. Lim, *Continued fraction absorbing boundary conditions for convex polygonal domains*, Int. J. Num. Meth. Engng., 66 (2006), pp. 949–977.

[70] T. Hagstrom, *Radiation boundary conditions for the numerical simulation of waves*, Acta Numerica, 8 (1999), pp. 47–106.

[71] ——, *New results on absorbing layers and radiation boundary conditions*, in Topics in Computational Wave Propagation, M. Ainsworth, P. Davies, D. Duncan, P. Martin, and B. Rynne, eds., Springer-Verlag, 2003, pp. 1–42.

[72] T. Hagstrom, D. Givoli, M. de Castro, and D. Tzemach, *Local high-order ABCs for time-dependent waves in guides*, J. Comput. Acoust., 15 (2007), pp. 1–22.

[73] T. Hagstrom, D. Givoli, and T. Warburton, *Radiation boundary conditions for time-dependent waves based on complete plane wave expansions.* To appear, 2008.

[74] T. Hagstrom and G. Hagstrom, *Grid stabilization of high-order one-sided differencing I: First order hyperbolic systems*, J. Comput. Phys., 223 (2007), pp. 316–340.

[75] T. Hagstrom and S. Lau, *Radiation boundary conditions for Maxwell's equations: a review of accurate time-domain formulations*, J. Comput. Math., 25 (2007), pp. 305–336.

[76] T. Hagstrom, A. Mar-Or, and D. Givoli, *High-order local absorbing boundary conditions for the wave equation: extensions and improvements*, J. Comput. Phys., 227 (2008), pp. 3322–3357.

[77] T. Hagstrom and T. Warburton, *High-order local radiation boundary conditions: Corner compatibility conditions and extensions to first order systems*, Wave Motion, 39 (2004), pp. 327–338.

[78] L. Halpern and L. Trefethen, *Wide-angle one-way wave equations*, J. Acoust. Soc. Am., 84 (1988), pp. 1397–1404.

[79] E. Heyman, *Time-dependent plane-wave spectrum representations for radiation from volume source distributions*, J. Math. Phys., 37 (1996), pp. 658–681.

[80] D. Justo, T. Warburton, and T. Hagstrom, *Solving scattering problems for maxwell's equations using polygonal artificial boundaries*, in 7th International Conference on Mathematical and Numerical Aspects of Wave Propagation Phenomena, 2005, pp. 71–73.

[81] H.-O. Kreiss and J. Lorenz, *Initial-Boundary Value Problems and the Navier-Stokes Equations*, Academic Press, New York, 1989.

[82] C. Lubich and A. Schädle, *Fast convolution for non-reflecting boundary conditions*, SIAM J. Sci. Comput., 24 (2002), pp. 161–182.

[83] D. Newman, *Rational approximation to* $|x|$, Michigan Math. J., 11 (1964), pp. 11–14.

[84] P. Petrushev and V. Popov, *Rational Approximation of Real Functions*, vol. 28 of Encyclopedia of Mathematics, Cambridge University Press, Cambridge, 1987.

[85] L. Trefethen and L. Halpern, *Well-posedness of one-way wave equations and absorbing boundary conditions*, Math. Comp., 47 (1986), pp. 421–435.

[86] O. Vacus, *Mathematical analysis of absorbing boundary conditions for the wave equation: The corner problem*, Math. Comput., 74 (2005), pp. 177–200.

[87] T. Warburton and T. Hagstrom, *Complete radiation boundary conditions for electromagnetic scattering.* Submitted, 2008.

[88] RJ Mailloux, SG Santarelli and TM Roberts, *Wideband arrays using irregular (polyomino) shaped subarrays*, Electr. Lett., Vol. 42, No 18, pp. 1019–1020, 2006.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* 03-12-2008	**2. REPORT TYPE** Final Technical Report
3. DATES COVERED *(From - To)*	01-07-2005 to 30-11-2008
4. TITLE AND SUBTITLE	Towards High Resolution Numerical Algorithms for Wave Dominated Physical Phenomena
5a. CONTRACT NUMBER	
5b. GRANT NUMBER	FA9550-05-1-0473
5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)	Timothy Warburton
5d. PROJECT NUMBER	
5e. TASK NUMBER	
5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	Department of Computational and Applied Mathematics Rice University, 6100 Main St Houston, TX 77005
8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)	Air Force Office of Scientific Research North Randolph Street Suite 325, Rm 3112 Arlington, VA 22203
10. SPONSOR/MONITOR'S ACRONYM(S)	AFOSR
11. SPONSOR/MONITOR'S REPORT NUMBER(S)	AFRL-OSR-VA-TR-2012-0488
12. DISTRIBUTION / AVAILABILITY STATEMENT	A - Approve For Public Release
13. SUPPLEMENTARY NOTES	

14. ABSTRACT

During this project several new numerical algorithms were designed, analyzed, and implemented with the goal of accelerating computational experiments in time-domain electromagnetics and acoustics. Novel results include a new family of radiation boundary conditions used for truncating computational domains used in external scattering simulations. A new filter and also local time stepping method were created for stabilizing coarse temporal approximations for discontinuous Galerkin time-domain computations.

In additional work a new tiling algorithm was developed allowing large numbers of polyomino tiles to be arranged to cover a two dimensional shape without leaving gaps. This abstract algorithm has application in the design of passive array antenna to minimize side-lobe radiation.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** / **19b. TELEPHONE NUMBER** *(include area code)*
				55	